

LNCS 3064

Daniel Bienstock
George Nemhauser (Eds.)

Integer Programming and Combinatorial Optimization

10th International IPCO Conference
New York, NY, USA, June 2004
Proceedings



Springer

Commenced Publication in 1973

Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

New York University, NY, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Daniel Bienstock George Nemhauser (Eds.)

Integer Programming and Combinatorial Optimization

10th International IPCO Conference
New York, NY, USA, June 7-11, 2004
Proceedings



Springer

Volume Editors

Daniel Bienstock
Columbia University, Department of IEOR
500 West 120th Street, New York, NY 10027, USA
E-mail: dano@columbia.edu

George Nemhauser
School of Industrial and Systems Engineering, Georgia Institute of Technology
Atlanta, GA 30332, USA
E-mail: george.nemhauser@isye.gatech.edu

Library of Congress Control Number: 2004106214

CR Subject Classification (1998): G.1.6, G.2.1, F.2.2, I.3.5

ISSN 0302-9743
ISBN 3-540-22113-1 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable to prosecution under the German Copyright Law.

Springer-Verlag is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2004
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 11008705 06/3142 5 4 3 2 1 0

Preface

This volume contains the papers accepted for publication at IPCO X, the Tenth International Conference on Integer Programming and Combinatorial Optimization, held in New York City, New York, USA, June 7–11, 2004. The IPCO series of conferences presents recent results in theory, computation and applications of integer programming and combinatorial optimization.

These conferences are sponsored by the Mathematical Programming Society, and are held in those years in which no International Symposium on Mathematical Programming takes place. IPCO VIII was held in Utrecht (The Netherlands) and IPCO IX was held in Cambridge (USA).

A total of 109 abstracts, mostly of very high quality, were submitted. The Program Committee accepted 32, in order to meet the goal of having three days of talks with no parallel sessions. Thus, many excellent abstracts could not be accepted.

The papers in this volume have not been refereed. It is expected that revised versions of the accepted papers will be submitted to standard scientific journals for publication.

The Program Committee thanks all authors of submitted manuscripts for their support of IPCO.

March 2004

George Nemhauser
Daniel Bienstock

Organization

IPCO X was hosted by the Computational Optimization Research Center (CORC), Columbia University.

Program Committee

Egon Balas
Daniel Bienstock
Robert E. Bixby
William Cook
Gerard Cornuéjols
William Cunningham
Bert Gerards
Ravi Kannan
George Nemhauser, Chair
William Pulleyblank
Laurence A. Wolsey

Organizing Committee

Daniel Bienstock, Chair
Garud Iyengar
Jay Sethuraman
Cliff Stein

Sponsoring Institutions

Bob and Betty Bixby
IBM
ILOG
The Fu Foundation School of Engineering and Applied Science, Columbia University
Mathematical Programming Society

Table of Contents

Session 1

- Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem 1
R. Fukasawa, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, R.F. Werneck

- Metric Inequalities and the Network Loading Problem 16
P. Avella, S. Mattia, A. Sassano

- Valid Inequalities Based on Simple Mixed-Integer Sets 33
S. Dash, O. Günlük

Session 2

- The Price of Anarchy when Costs Are Non-separable and Asymmetric 46
G. Perakis

- Computational Complexity, Fairness, and the Price of Anarchy of the Maximum Latency Problem 59
J.R. Correa, A.S. Schulz, N.E. Stier Moses

- Polynomial Time Algorithm for Determining Optimal Strategies in Cyclic Games 74
D. Lozovanu

Session 3

- A Robust Optimization Approach to Supply Chain Management 86
D. Bertsimas, A. Thiele

- Hedging Uncertainty: Approximation Algorithms for Stochastic Optimization Problems 101
R. Ravi, A. Sinha

- Scheduling an Industrial Production Facility 116
E. Asgeirsson, J. Berry, C.A. Phillips, D.J. Phillips, C. Stein, J. Wein

Session 4

- Three Min-Max Theorems Concerning Cyclic Orders of Strong Digraphs . 132
S. Bessy, S. Thomassé

A TDI Description of Restricted 2-Matching Polytopes	139
<i>G. Pap</i>	

Enumerating Minimal Dicuts and Strongly Connected Subgraphs and Related Geometric Problems	152
<i>E. Boros, K. Elbassioni, V. Gurvich, L. Khachiyan</i>	

Session 5

Semi-continuous Cuts for Mixed-Integer Programming	163
<i>I.R. de Farias Jr.</i>	

Combinatorial Benders' Cuts	178
<i>G. Codato, M. Fischetti</i>	

A Faster Exact Separation Algorithm for Blossom Inequalities	196
<i>A.N. Letchford, G. Reinelt, D.O. Theis</i>	

Session 6

LP-based Approximation Algorithms for Capacitated Facility Location	206
<i>R. Levi, D.B. Shmoys, C. Swamy</i>	

A Multi-exchange Local Search Algorithm for the Capacitated Facility Location Problem	219
<i>J. Zhang, B. Chen, Y. Ye</i>	

Separable Concave Optimization Approximately Equals Piecewise Linear Optimization	234
<i>T.L. Magnanti, D. Stratila</i>	

Session 7

Three Kinds of Integer Programming Algorithms Based on Barvinok's Rational Functions	244
<i>J.A. De Loera, D. Haws, R. Hemmecke, P. Huggins, R. Yoshida</i>	

The Path-Packing Structure of Graphs	256
<i>A. Sebő, L. Szegő</i>	

More on a Binary-Encoded Coloring Formulation	271
<i>J. Lee, F. Margot</i>	

Session 8

Single Machine Scheduling with Precedence Constraints	283
<i>J.R. Correa, A.S. Schulz</i>	

The Constrained Minimum Weighted Sum of Job Completion Times Problem	298
<i>A. Levin, G.J. Woeginger</i>	

Session 9

Near-Optimum Global Routing with Coupling, Delay Bounds, and Power Consumption	308
<i>J. Vygen</i>	

A Flow-Based Method for Improving the Expansion or Conductance of Graph Cuts	325
<i>K. Lang, S. Rao</i>	

All Rational Polytopes Are Transportation Polytopes and All Polytopal Integer Sets Are Contingency Tables	338
<i>J. De Loera, S. Onn</i>	

Session 10

A Capacity Scaling Algorithm for M-convex Submodular Flow	352
<i>S. Iwata, S. Moriguchi, K. Murota</i>	

Integer Concave Cocirculations and Honeycombs	368
<i>A.V. Karzanov</i>	

Minsquare Factors and Maxfix Covers of Graphs	388
<i>N. Apollonio, A. Sebő</i>	

Session 11

Low-Dimensional Faces of Random 0/1-Polytopes	401
<i>V. Kaibel</i>	

On Polyhedra Related to Even Factors	416
<i>T. Király, M. Makai</i>	

Optimizing over Semimetric Polytopes	431
<i>A. Frangioni, A. Lodi, G. Rinaldi</i>	

Author Index	445
---------------------------	-----

Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem

Ricardo Fukasawa¹, Jens Lysgaard², Marcus Poggi de Aragão³, Marcelo Reis³,
Eduardo Uchoa^{4*}, and Renato F. Werneck⁵

¹ School of Industrial and Systems Engineering, GeorgiaTech, USA
rfukasaw@isy.e.gatech.edu.

² Department of Management Science and Logistics,
Aarhus School of Business, Denmark
lys@asb.dk

³ Departamento de Informática, PUC Rio de Janeiro, Brazil
{poggi,mreis}@inf.puc-rio.br

⁴ Departamento de Engenharia de Produção,
Universidade Federal Fluminense, Brazil.
uchoa@producao.uff.br

⁵ Department of Computer Science, Princeton University, USA
rwerneck@cs.princeton.edu

Abstract. The best exact algorithms for the Capacitated Vehicle Routing Problem (CVRP) have been based on either branch-and-cut or Lagrangean relaxation/column generation. This paper presents an algorithm that combines both approaches: it works over the intersection of two polytopes, one associated with a traditional Lagrangean relaxation over q -routes, the other defined by bound, degree and capacity constraints. This is equivalent to a linear program with exponentially many variables and constraints that can lead to lower bounds that are superior to those given by previous methods. The resulting branch-and-cut-and-price algorithm can solve to optimality all instances from the literature with up to 135 vertices. This doubles the size of the instances that can be consistently solved.

1 Introduction

Let $G = (V, E)$ be an undirected graph with vertices $V = \{0, 1, \dots, n\}$. Vertex 0 represents the *depot*, whereas all others represent *clients*, each with an associated demand $d(\cdot)$. Each edge $e \in E$ has a nonnegative length $\ell(e)$. Given G and two positive integers (K and C), the *Capacitated Vehicle Routing Problem* (CVRP) consists of finding routes for K vehicles satisfying the following constraints: (i) each route starts and ends at the depot, (ii) each client is visited by a single vehicle, and (iii) the total demand of all clients in any route is at most C . The goal is to minimize the sum of the lengths of all routes. This classical NP-hard problem is a natural generalization of the Travelling Salesman Problem (TSP),

* Corresponding author.

and has widespread application itself. The CVRP was first proposed in 1959 by Dantzig and Ramser [13] and has received close attention from the optimization community since then.

A landmark exact algorithm for the CVRP, presented in 1981 by Christofides, Mingozzi and Toth [11], uses a Lagrangean bound from minimum q -route subproblems. A q -route is a walk that starts at the depot, traverses a sequence of clients with total demand at most C , and returns to the depot. Some clients may be visited more than once, so the set of valid CVRP routes is strictly contained in the set of q -routes. The resulting branch-and-bound algorithm could solve instances with up to 25 vertices, a respectful size at the time.

Several other algorithms using Lagrangean relaxation appear in the literature. Christofides et al. [11] also describe a lower bound based on k -degree center trees, which are minimum spanning trees having degree $K \leq k \leq 2K$ on the depot, plus $2K - k$ least-cost edges. Lagrangean bounds based on K -trees (sets of $n + K - 1$ edges spanning V) having degree $2K$ in the depot were used by Fisher [14] and by Martinhon, Lucena, and Maculan [24], among others. Miller [25] presented an algorithm based on minimum b -matchings having degree $2K$ at the depot and 2 on the remaining vertices. Lagrangean bounds can be improved by dualizing capacity inequalities [14,25] and also comb and multistar inequalities [24].

Another family of exact algorithms stems from the formulation of the CVRP as a set partitioning problem by Balinski and Quandt [8]. A column covers a set of vertices S with total demand not exceeding C and has the cost of a minimum route over $\{0\} \cup S$. Unfortunately, the formulation is not practical because pricing over the exponential number of columns requires the solution of capacitated prize-collecting TSPs, a problem almost as difficult as the CVRP itself. Agarwal, Marthur and Salkin [7] proposed a column generation algorithm on a modified set partitioning problem where column costs are given by a linear function over the vertices yielding a lower bound on the actual route cost. Columns with the modified cost can be priced by solving easy knapsack problems. Hadjiconstantinou et al. [17] derive lower bounds from heuristic solutions to the dual of the set partitioning formulation. The dual solutions are obtained by the so-called additive approach, combining the q -route and k -shortest path relaxations.

For further information and comparative results on the algorithms mentioned above, we refer the reader to the surveys by Toth and Vigo [31,32].

Recent research on the CVRP has been concentrated on the polyhedral description of the convex hull of the edge incidence vectors that correspond to K feasible routes and on the development of effective separation procedures [1,3,5,6,12,20,26]. In particular, Araque et al. [4], Augerat et al. [6], Blasum and Hochstättler [9], Ralphs et al. [30], Achuthan, Caccetta, and Hill [2] and Lysgaard, Letchford, and Eglese [23] describe complete branch-and-cut (BC) algorithms. These are the best exact methods currently available for the CVRP. However, the addition of several elaborate classes of cuts does not guarantee tight lower bounds, especially for large values of K ($K \geq 7$, say). Closing the resulting duality gap usually requires exploring several nodes in the branch-and-cut tree.

Even resorting to massive computational power (up to 80 processors running in parallel in a recent work by Ralphs [29,30]) several instances with fewer than 80 vertices, including some proposed more than 30 years ago by Christofides and Eilon [10], can not be solved at all. In fact, branch-and-cut algorithms for the CVRP seem to be experiencing a “diminishing returns” stage, where substantial theoretical and implementation efforts achieve practical results that are only marginally better than those of previous works.

We present a new exact algorithm for the CVRP that seems to break through this situation. The main idea is to combine the branch-and-cut approach with the q -routes approach (which we interpret as column generation instead of the original Lagrangean relaxation) to derive superior lower bounds. Since the resulting formulation has an exponential number of both columns and rows, this leads to a branch-and-cut-and-price (BCP) algorithm. Computational experiments over the main instances from the literature show that this algorithm is very consistent on solving instances with up to 100 vertices. Eighteen open instances were solved for the first time.

The idea of combining column and cut generation to improve lower bounds has rarely been used, since new dual variables corresponding to separated cuts may have the undesirable effect of changing the structure of the pricing subproblem. However, if cuts are expressed in terms of variables from a suitable original formulation, they can be incorporated into the column generation process without disturbing the pricing. We refer to branch-and-bound procedures based on such formulations as *robust branch-and-cut-and-price* algorithms. Poggi de Aragão and Uchoa [28] present a detailed discussion on this subject, including new reformulation techniques that extend the applicability of robust branch-and-cut-and-price algorithms to virtually any combinatorial optimization problem. This article on the CVRP is part of a larger effort to demonstrate that these methods lead to significant improvements on a wide variety of problems. Major advances have already been reported on two other problems: capacitated minimum spanning tree [15] and generalized assignment [27].

This article is organized as follows. Section 2 describes the integer programming formulation we will deal with. Section 3 gives a general description of our algorithm, including its two main components: column and cut generation. Following the work of Irnich and Villeneuve [18] on the CVRP with time windows, our column generation procedure eliminates q -routes with small cycles. The separation routines are based on the families of inequalities recently discussed by Letchford, Eglese, and Lysgaard [20,23]. Section 4 presents an empirical analysis of our method. Final remarks are made in Sect. 5.

2 The New Formulation

A classical formulation for the CVRP [19] represents by x_{ij} the number of times a vehicle traverses the edge $(i, j) \in E$. The set of client vertices is denoted by $V_+ = \{1, \dots, n\}$. Given a set $S \subseteq V_+$, let $d(S)$ be the sum of the demands of all vertices in S , and let $\delta(S)$ be the cut-set defined by S . Also, let $k(S) = \lceil d(S)/C \rceil$. Define the following polytope in $\mathbb{R}^{|E|}$:

$$P_1 = \begin{cases} \sum_{e \in \delta(\{i\})} x_e = 2 & \forall i \in V_+ \\ \sum_{e \in \delta(\{0\})} x_e = 2 \cdot K & (2) \\ \sum_{e \in \delta(S)} x_e \geq 2 \cdot k(S) & \forall S \subseteq V_+ \\ x_e \leq 1 & \forall e \in E \setminus \delta(\{0\}) \\ x_e \geq 0 & \forall e \in E . \end{cases}$$

Constraints (1) state that each client is visited once by some vehicle, whereas (2) states that K vehicles must leave and enter the depot. Constraints (3) are rounded capacity inequalities, which require all subsets to be served by enough vehicles. Constraints (4) enforce that each edge not adjacent to the depot is traversed at most once (edges adjacent to the depot can be used twice when a route serves only one client). The integer vectors x in P_1 define all feasible solutions for the CVRP. There are exponentially many inequalities of type (3), so the lower bound given by

$$L_1 = \min_{x \in P_1} \sum_{e \in E} \ell_e x_e$$

must be computed by a cutting plane algorithm.

Alternatively, a formulation with an exponential number of columns can be obtained by defining variables (columns) that correspond to q -routes without 2-cycles (subpaths $i \rightarrow j \rightarrow i$, $i \neq 0$). Restricting the q -routes to those without such cycles improves the formulation and does not change the complexity of the pricing [11]. Let Q be an $m \times p$ matrix where the columns are the edge incidence vectors of all p such q -routes. Let q_j^e be the coefficient associated with edge e in the j -th column of Q . Consider the following polytope in $\mathbb{R}^{|E|}$, defined as the projection of a polytope in $\mathbb{R}^{p+|E|}$:

$$P_2 = \text{proj}_x \begin{cases} \sum_{j=1}^p q_j^e \cdot \lambda_j - x_e = 0 & \forall e \in E \\ \sum_{j=1}^p \lambda_j = K \\ \sum_{e \in \delta(\{i\})} x_e = 2 & \forall i \in V_+ \\ x_e \geq 0 & \forall e \in E \\ \lambda_j \geq 0 & \forall j \in \{1, \dots, p\} . \end{cases}$$

Constraints (5) define the coupling between variables x and λ . Constraint (6) defines the number of vehicles to use. It can be shown that the set of integer vectors in P_2 also defines all feasible solutions for the CVRP. Due to the exponential number of variables λ , the lower bound given by

$$L_2 = \min_{x \in P_2} \sum_{e \in E} \ell_e x_e$$

must be computed using column generation or Lagrangean relaxation.

The description of polyhedra associated with column generation or Lagrangian relaxation in terms of two sets of variables, λ and x , used in the definition of P_2 , is called *Explicit Master* in [28]. The main contribution of this article is a formulation that amounts to optimizing over the intersection of polytopes P_1 and P_2 . The Explicit Master format describes such intersection as follows:

$$P_3 = P_1 \cap P_2 = \text{proj}_x \left\{ \begin{array}{ll} \sum_{e \in \delta(\{i\})} x_e = 2 & \forall i \in V_+ \\ \sum_{e \in \delta(\{0\})} x_e = 2 \cdot K & (2) \\ \sum_{e \in \delta(S)} x_e \geq 2 \cdot k(S) & \forall S \subseteq V_+ \\ x_e \leq 1 & \forall e \in E \setminus \delta(\{0\}) \\ \sum_{j=1}^p q_j^e \cdot \lambda_j - x_e = 0 & \forall e \in E \\ \sum_{j=1}^p \lambda_j = K & (6) \\ x_e \geq 0 & \forall e \in E \\ \lambda_j \geq 0 & \forall j \in \{1, \dots, p\} . \end{array} \right. \quad (1)$$

Constraint (6) can be discarded, since it is implied by (2) and (5). Computing the improved lower bound

$$L_3 = \min_{x \in P_3} \sum_{e \in E} \ell_e x_e$$

requires solving a linear program with an exponential number of both variables and constraints. A more compact LP is obtained if every occurrence x_e in (1)–(4) is replaced by its equivalent given by (5). The resulting LP will be referred to as the *Dantzig-Wolfe Master problem* (DWM):

$$\text{DWM} = \left\{ \begin{array}{ll} L_3 = \min \sum_{j=1}^p \sum_{e \in E} \ell_e \cdot q_j^e \cdot \lambda_j & (7) \\ \text{s.t.} \quad \sum_{j=1}^p \sum_{e \in \delta(\{i\})} q_j^e \cdot \lambda_j = 2 & \forall i \in V_+ \\ \sum_{j=1}^p \sum_{e \in \delta(\{0\})} q_j^e \cdot \lambda_j = 2 \cdot K & (9) \\ \sum_{j=1}^p \sum_{e \in \delta(S)} q_j^e \cdot \lambda_j \geq 2 \cdot k(S) & \forall S \subseteq V_+ \\ \sum_{j=1}^p q_j^e \cdot \lambda_j \leq 1 & \forall e \in E \setminus \delta(\{0\}) \\ \lambda_j \geq 0 & \forall j \in \{1, \dots, p\} . \end{array} \right. \quad (8)$$

Capacity inequalities are not the only ones that can appear in the DWM. A generic cut $\sum_{e \in E} a_e x_e \geq b$ can be included as $\sum_{j=1}^p (\sum_{e \in E} a_e q_j^e) \cdot \lambda_j \geq b$. In fact, we added all classes of cuts described in [23].

3 The Branch-and-Cut-and-Price Algorithm

Column Generation. The reduced cost of a column (λ variable) is the sum of the reduced costs of the edges in the corresponding q -route. Let μ , ν , π , and ω be the dual variables associated with constraints (8), (9), (10), and (11), respectively. The reduced cost \bar{c}_e of an edge e is given by:

$$\bar{c}_e = \begin{cases} \ell_e - \mu_i - \mu_j - \sum_{S|\delta(S) \ni e} \pi_S - \omega_e & e = \{i, j\} \in E \setminus \delta(\{0\}) \\ \ell_e - \nu - \mu_j - \sum_{S|\delta(S) \ni e} \pi_S & e = \{0, j\} \in \delta(\{0\}) . \end{cases}$$

An additional generic cut $\sum_{j=1}^p (\sum_{e \in E} a_e q_j^e) \cdot \lambda_j \geq b$ in DWM, with dual variable α , contributes with the value $-a_e \cdot \alpha$ to the computation of \bar{c}_e .

The pricing subproblem consists of finding q -routes (columns) of minimum reduced cost. Although this problem is NP-hard in general, it can be solved in pseudopolynomial time if all demands are integer [11].

The basic data structure is a $C \times n$ matrix M . Each entry $M(d, v)$ will represent the least costly walk that reaches vertex v using total demand exactly d . The entry contains a *label* consisting of the vertex (v), the cost of the walk, and a pointer to a label representing the walk up to the previous vertex. The matrix is filled by dynamic programming, row by row, starting with $d = 1$. For each row d , the algorithm goes through each entry v and, for each neighbor w of v , evaluates the extension of the walk represented by $M(d, v)$ to w . If $\bar{c}(M(d, v)) + \bar{c}_{(v,w)} < \bar{c}(M(d+d(w), w))$, $M(d+d(w), w)$ is updated. Eventually, we will have the most negative walk with accumulated demand at most C that arrives at each vertex v . Extending the walk to the depot (whose demand is zero), we obtain the corresponding q -route. All negative q -routes thus found (there will be at most n) are added to the linear program. There are nC entries in the matrix, and each is processed in $O(n)$ time, so the total running time is $O(n^2 C)$.

To strengthen the formulation, we look for s -cycle-free q -routes, for small values of s . The algorithm operates as above, using dynamic programming to fill a $C \times n$ matrix with partial walks. Each entry in the matrix is no longer a single label, but a *bucket* of labels. Bucket $M(d, v)$ represents not only the cheapest s -cycle-free walk with total demand d that ends in v , but also alternative walks that ensure that all possible s -vertex extensions from v are considered. Deciding which walks to keep is trivial for $s = 2$ [11], but quite intricate for $s \geq 3$. We handle this with the approach originally proposed by Irnich and Villeneuve [18] (the reader is referred to their paper for details). They show that buckets must have size at least $s!$, so the method quickly becomes impractical.

We use three heuristics to accelerate the algorithm. They all find only s -cycle-free q -routes, but not necessarily the minimum one. Hence, whenever they find no negative route, the exact algorithm must verify that none exists.

The first acceleration technique we use is *scaling*, which considers $g > 1$ units of demand at a time. The exact algorithm is run with a modified demand $d'_v(g) = \lceil d_v/g \rceil$ for every vertex v , and modified capacity $C' = \lfloor C/g \rfloor$ for the

vehicles. The running time will be proportional to C' instead of C , but the solutions found will still be valid in the original setting.

The second technique is *sparsification*. Intuitively, small edges in the original graph are more likely to appear in the solution. With that in mind, we pre-compute a set of five disjoint spanning forests using an extension of Kruskal's algorithm similar to the “edge-greedy heuristic” described in [34]. By considering only edges in those trees, the dynamic programming will be limited to a restricted set of neighbors when processing each vertex.

The third acceleration is *bucket pruning*, which consists of storing only s (instead of $s!$) labels per bucket. To minimize the number of valid s -extensions blocked, we keep only labels that differ significantly from each other.

The algorithm starts using all three acceleration techniques. Whenever it fails to find a negative q -route, it turns one of them off. If the algorithm fails with no acceleration, we can safely declare that no negative q -route exists. On the other hand, if only a subset of the heuristics is in use and they do find a negative q -route, other heuristics are turned back on.

Cut Generation. At first, the LP formulation includes only degree constraints; violated cuts are added as needed. Besides the well-known rounded capacity cuts (10) and bound cuts (11), we also use framed capacity, strengthened comb, multistar, partial multistar, generalized multistar and hypotour cuts, all presented in [23]. We use the CVRPSEP package [22] to separate those cuts. Since they are defined in terms of the x_e^* variables, not readily available in the DWM formulation, we convert λ_j^* variables into the corresponding x_e^* variables as needed.

Branching Rule. The algorithm starts by adding columns to the formulation until there are none with negative reduced costs, then it looks for all violated cuts. This is repeated until both column generation and separation fail. If the node cannot be fathomed at this point, we branch.

The branching rule is an adaptation of the rule used in [23]. We choose as branching candidates sets S such that $2 < x^*(\delta(S)) < 4$ and branch by imposing the disjunction $(x(\delta(S)) = 2) \vee (x(\delta(S)) \geq 4)$.

We use strong branching to select which set to branch on. Since computing the actual lower bound for each child node can be very time-consuming, we estimate it by performing a small number of column generation iterations. Only p candidate sets ($5 \leq p \leq \max\{10 - \text{depth}, 5\}$) are evaluated. This limits the time spent on strong branching, especially when the depth is high. Priority is given to sets with smaller values of $|x^*(\delta(S)) - 2.7|/d(S)$. We use 2.7 because constraint $x(\delta(S)) = 2$ tends to have a greater impact on the LP relaxation than $x(\delta(S)) \geq 4$, leading to an imbalance in the branch-and-bound tree. Values closer to 2 than 4 increase the impact of imposing $x(\delta(S)) \geq 4$.

Node Selection and Primal Bounds. We used as initial primal bound the best previously known solution plus one. The node selection strategy chosen was depth first search, because it requires less memory.

Dynamic Choice of Column Generation. It usually pays off to combine the strengths of cut and column generation into a single algorithm. However, in some instances, the increase in the lower bound is not worth the additional time spent. In these cases pure branch-and-cut performs better than branch-and-cut-and-price. We modified our algorithm to handle such cases. In the root node, we start as in a pure branch-and-cut, without any column generation. After that, the root node is solved again with a branch-and-cut-and-price using the column generation with $s = 2$. Then we solve it once again, changing s to 3. For the rest of the branch-and-bound tree, the algorithm picks the strategy with the best balance between running time and bound quality.

4 Computational Experiments

We tested our algorithm on most instances available at www.branchandcut.org. All instances with up to 135 vertices were solved to optimality, eighteen of them for the first time. The tests were run on a Pentium IV, 2.4 GHz, with 1GB of RAM. Throughout this section, we will refer to the branch-and-cut-and-price algorithm with dynamic choice of column generation as *Dyn-BCP* and the branch-and-cut-and-price with pre-determined column generation as *BCP*.

Tables 1 and 2 detail the results obtained by Dyn-BCP. Columns **Root LB** and **Root Time** give the lower bound and total CPU time of the root node of the branch-and-bound tree. The running time includes the dynamic choice of column generation and strong branching. The next column represents the size s of the cycles eliminated by the column generation procedure (“–” indicates that column generation was not used). **Tree Size** represents the total number of branch-and-bound nodes explored, and **Total Time** is the total CPU time spent by the algorithm. Finally, **Prev UB** and **Our UB** indicate the best solution value available in the literature and the one obtained by our procedure. Values in bold indicate proven optimality.

Table 3 compares several different lower bounds obtained for the CVRP. Lower bounds L1, L2 and L3 (defined in Sect. 2) are presented in the first three columns. Column **LLE03** shows the lower bounds obtained by [23], which are the best available in the literature. The next three columns contain the lower bounds obtained at the root node of the BCP with s -cycle elimination (for $s = 2, 3, 4$). Column **OPT** is the optimal solution value and **T(3)** is the total CPU time (in seconds) spent at the root node of BCP for $s = 3$, the value most commonly used. On average, the algorithm for $s = 4$ was 61% slower than for $s = 3$, which in turn was 12% slower than for $s = 2$. The last line shows the average gap of each bound, considering only instances with a known optimum.

Table 3 shows that major improvements can be obtained by using the BCP approach instead of pure branch-and-cut. Also, eliminating longer cycles often increases the lower bounds significantly.

The greatest improvements usually happen when the ratio n/K is smallest. In such cases, columns without small cycles are very close to being actual routes, so one can expect the duality gap to be reduced. Figure 1, created from the instances

Table 1. Results of the Dyn-BCP algorithm for the A and B instances.

Instance	Root LB	Root Time (s)	s	Tree Size	Total Time (s)	Prev. UB	Our UB
A-n37-k5	664.8	16	—	8	19	669	669
A-n37-k6	932.6	30	3	74	379	949	949
A-n38-k5	716.5	12	—	52	26	730	730
A-n39-k5	816.6	107	3	11	167	822	822
A-n39-k6	822.8	39	3	11	98	831	831
A-n44-k6	934.8	52	2	6	90	937	937
A-n45-k6	938.1	52	3	11	170	944	944
A-n45-k7	1139.3	88	3	26	331	1146	1146
A-n46-k7	914.0	63	2	3	92	914	914
A-n48-k7	1069.1	72	3	8	166	1073	1073
A-n53-k7	1003.9	138	3	16	611	1010	1010
A-n54-k7	1153.9	125	3	90	1409	1167	1167
A-n55-k9	1067.4	32	3	7	84	1073	1073
A-n60-k9	1344.4	161	3	224	3080	1354	1354
A-n61-k9	1022.5	108	3	121	1883	1034	1034
A-n62-k8	1280.4	722	3	101	3102	1290	1288
A-n63-k9	1607.0	238	3	49	1046	1616	1616
A-n63-k10	1299.1	136	3	387	4988	1315	1314
A-n64-k9	1385.3	265	3	648	11254	1402	1401
A-n65-k9	1166.5	154	3	17	516	1174	1174
A-n69-k9	1141.4	289	3	391	7171	1159	1159
A-n80-k10	1754.0	1120	3	153	6464	1763	1763
B-n38-k6	800.2	10	—	14	14	805	805
B-n39-k5	549.0	3	—	1	3	549	549
B-n41-k6	826.4	13	—	8	18	829	829
B-n43-k6	733.7	13	—	74	29	742	742
B-n44-k7	909.0	9	—	1	9	909	909
B-n45-k5	747.5	10	—	19	16	751	751
B-n45-k6	677.5	224	3	3	279	678	678
B-n50-k7	741.0	5	—	3	6	741	741
B-n50-k8	1295.0	97	3	287	2845	1312	1312
B-n51-k7	1025.2	16	—	83	46	1032	1032
B-n52-k7	745.8	7	—	9	9	747	747
B-n56-k7	704.0	15	—	9	22	707	707
B-n57-k7	1149.2	76	—	133	168	1153	1153
B-n57-k9	1596.0	61	3	15	193	1598	1598
B-n63-k10	1479.4	231	—	501	682	1496	1496
B-n64-k9	859.3	70	—	7	86	861	861
B-n66-k9	1307.5	145	3	126	1778	1316	1316
B-n67-k10	1024.4	218	—	327	568	1032	1032
B-n68-k9	1263.0	260	3	5912	87436	1275*	1272
B-n78-k10	1215.2	193	3	90	1053	1221	1221

* An earlier version of our algorithm [16] found a solution of 1272 but could not prove its optimality. Using that information, K. Wenger [33] solved this instance.

Table 2. Results of the Dyn-BCP algorithm for the E, F, M and P instances.

Instance	Root LB	Root Time (s)	<i>s</i>	Tree Size	Total Time (s)	Prev. UB	Our UB
E-n13-k4	247.0	0	—	1	0	247	247
E-n22-k4	375.0	0	—	1	0	375	375
E-n23-k3	569.0	0	—	1	0	569	569
E-n30-k3	533.3	7	—	6	7	534	534
E-n31-k7	378.5	4	2	2	6	379	379
E-n33-k4	834.5	14	—	5	15	835	835
E-n51-k5	518.2	51	—	8	65	521	521
E-n76-k7	670.0	264	2	1712	46520	682	682
E-n76-k8	726.5	277	2	1031	22891	735	735
E-n76-k10	817.4	354	3	4292	80722	830	830
E-n76-k14	1006.5	224	3	6678	48637	1021	1021
E-n101-k8	805.2	1068	3	11622	801963	815	815
E-n101-k14	1053.8	658	3	5848	116284	1071	1067
F-n45-k4	724.0	8	—	1	8	724	724
F-n72-k4	236.4	70	—	42	121	237	237
F-n135-k7	1159.9	6618	—	25	7065	1162	1162
M-n101-k10	820.0	119	—	1	119	820	820
M-n121-k7	1031.1	5594	3	40	25678	1034	1034
P-n16-k8	449.0	1	2	3	1	450	450
P-n19-k2	212.0	1	—	1	1	212	212
P-n20-k2	213.0	1	—	9	1	216	216
P-n21-k2	211.0	1	—	1	1	211	211
P-n22-k2	216.0	2	—	2	2	216	216
P-n22-k8	603.0	3	2	1	3	603	603
P-n23-k8	529.0	18	2	1	18	529	529
P-n40-k5	456.9	28	—	5	34	458	458
P-n45-k5	506.6	59	3	11	194	510	510
P-n50-k7	551.5	79	3	7	143	554	554
P-n50-k8	616.3	102	3	1084	9272	649	631
P-n50-k10	689.3	50	3	65	304	696	696
P-n51-k10	735.2	35	3	22	105	741	741
P-n55-k7	557.9	90	2	450	4649	568	568
P-n55-k8	579.8	42	2	196	1822	588	588
P-n55-k10	681.4	107	3	1556	9076	699	694
P-n55-k15	972.8	251	3	398	1944	993	989
P-n60-k10	738.9	126	3	52	570	756	744
P-n60-k15	962.8	118	3	76	442	1033	968
P-n65-k10	786.0	159	3	23	422	792	792
P-n70-k10	814.5	292	3	1752	24039	834	827
P-n76-k4	588.8	363	—	59	572	593	593
P-n76-k5	616.8	273	—	3399	14546	627	627
P-n101-k4	678.5	1055	—	23	1253	681	681

Table 3. Comparison of lower bounds.

Instance	Lower Bounds							OPT	T(3)
	L1	L2	L3	LLE03	$s = 2$	$s = 3$	$s = 4$		
A-n53-k7	996.6	978.5	1002.2	998.7	1001.9	1003.8	1004.1	1010	29
A-n54-k7	1130.7	1114.0	1150.0	1135.3	1150.9	1153.6	1156.1	1167	26
A-n55-k9	1055.9	1025.4	1066.4	1058.3	1066.4	1067.3	1067.6	1073	12
A-n60-k9	1316.5	1305.6	1341.6	1319.6	1341.6	1344.4	1345.3	1354	27
A-n61-k9	1004.8	996.8	1018.6	1010.2	1018.9	1022.7	1023.3	1034	19
A-n62-k8	1244.1	1222.7	1274.1	1251.7	1273.7	1280.5	1281.1	1288	36
A-n63-k9	1572.2	1564.8	1603.5	1580.7	1603.6	1607.0	1608.9	1616	34
A-n63-k10	1262.2	1267.4	1294.5	1266.6	1294.4	1299.1	1302.5	1314	28
A-n64-k9	1340.1	1353.3	1378.9	1351.6	1379.0	1385.3	1389.2	1401	39
A-n65-k9	1151.1	1133.0	1163.4	1155.2	1164.4	1166.5	1168.3	1174	31
A-n69-k9	1108.9	1113.2	1138.4	1114.4	1139.3	1141.4	1144.1	1159	87
A-n80-k10	1699.9	1712.2	1749.7	1709.6	1749.6	1753.9	1755.6	1763	940
B-n50-k7	740.0	664.8	741.0	741.0	741.0	741.0	741.0	741	8
B-n50-k8	1279.2	1217.5	1291.8	1281.1	1291.8	1295.2	1302.0	1312	20
B-n51-k7	1024.6	918.4	1025.9	1025.6	1026.0	1026.4	1026.6	1032	21
B-n52-k7	745.0	640.2	746.3	746.0	746.4	746.4	746.4	747	17
B-n56-k7	703.4	606.9	704.5	705.0	704.5	705.0	705.0	707	19
B-n57-k7	1148.6	1058.5	1150.9	1150.1	1151.1	1151.6	1152.3	1153	126
B-n57-k9	1586.7	1511.5	1595.2	1589.2	1595.2	1595.8	1596.1	1598	16
B-n63-k10	1478.9	1418.4	1484.2	1481.0	1484.3	1487.2	1487.3	1496	32
B-n64-k9	858.5	769.3	860.1	860.5	860.9	861.0	860.5	861	44
B-n66-k9	1295.2	1223.1	1302.6	1298.5	1303.7	1307.5	1308.5	1316	58
B-n67-k10	1023.8	984.5	1026.4	1024.8	1026.4	1026.9	1027.2	1032	25
B-n68-k9	1256.8	1163.9	1261.5	1258.1	1261.6	1262.9	1263.5	1272	42
B-n78-k10	1202.3	1124.5	1212.5	1205.6	1212.7	1214.8	1215.7	1221	63
E-n51-k5	514.5	512.9	518.0	519.0	519.1	519.1	519.4	521	23
E-n76-k7	661.4	663.3	668.4	666.4	670.3	670.8	670.9	682	73
E-n76-k8	711.2	716.7	725.1	717.9	726.5	726.8	727.0	735	72
E-n76-k10	789.5	811.4	816.5	799.9	817.3	817.4	817.7	830	36
E-n76-k14	948.1	999.6	1004.8	969.6	1004.9	1006.5	1007.5	1021	15
E-n101-k8	796.4	786.4	801.8	802.6	804.6	805.2	805.2	815	320
E-n101-k14	1008.3	1045.1	1051.6	1026.9	1052.2	1053.9	1054.3	1067	69
M-n101-k10	819.5	798.1	820.0	820.0	820.0	820.0	820.0	820	31
M-n121-k7	1009.7	1013.0	1030.9	1017.4	1031.2	1031.8	1032.0	1034	1257
P-n50-k8	596.9	612.5	615.3	602.1	615.8	616.3	617.1	631	13
P-n55-k10	646.7	677.2	680.1	662.1	680.2	681.4	681.4	694	6
P-n55-k15	895.1	963.0	967.5	906.7	967.7	973.1	973.2	989	7
P-n60-k10	708.3	733.5	737.2	718.4	737.7	739.0	739.3	744	8
P-n60-k15	903.3	955.6	961.2	929.8	961.7	962.9	963.5	968	4
P-n65-k10	756.5	779.8	785.2	767.1	785.6	786.0	787.1	792	13
P-n70-k10	786.9	808.3	812.7	795.6	813.6	814.6	814.9	827	23
AvgGAP	2.92%	4.76%	1.02%	2.26%	0.97%	0.82%	0.74%	—	—

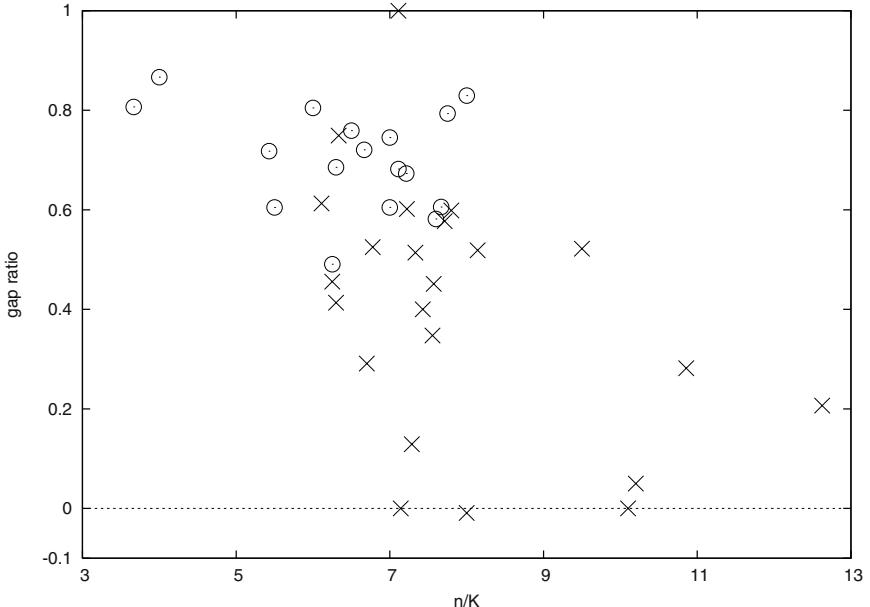


Fig. 1. Gap improvement of BCP with $s = 3$ (*s3*) with respect to [23] (*lle*). The graph represents $\frac{s3-lle}{opt-lle}$ as a function of $\frac{n}{K}$. Circles represent instances solved to optimality for the first time and crosses the ones already closed.

of Table 3 with a known optimal solution, illustrates this observation. The horizontal axis represents the ratio n/K . The value plotted is $(s3 - lle)/(opt - lle)$, where $s3$ is the bound of the BCP with $s = 3$, lle is the LLE03 bound and opt is the optimal solution value. This ratio represents the reduction of the duality gap relative to the LLE03 gap. The value is zero whenever both methods produce the same result; positive values mean that BCP gaps are smaller, and negative values favor LLE03. Circles represent instances that were solved by our algorithm for the first time, while crosses indicate instances that were already closed. Only one instance (B-n56-k7) had a negative value (-0.009). The gap was often reduced by more than 50%, enough to close several open instances.

5 Conclusion

This paper has shown that a careful combination of column and cut generation leads to a very effective algorithm for the CVRP. We consider the three main aspects that can lead to further improvements: cut generation, column generation, and the integration between them.

Consider cutting planes. Rounded capacity cuts are highly effective in our BCP—the efficient heuristic separation for them contributes significantly to the good overall performance. We did try all other families of cuts for the CVRP

known to be effective in a pure BC: framed capacities, generalized capacities, strengthened combs, multistars, and extended hypotours. Surprisingly, however, their effect was quite modest in practice—only on E-n101-k8 the root bound increased by more than two units. Any improvement to the cutting part of our BCP will therefore require new families of inequalities. Unfortunately, looking for facet-defining inequalities may not be enough within the BCP framework, since a facet of the CVRP polyhedron can also be a facet of the polyhedron induced by column generation over the q -routes.⁶ A new family of inequalities will only be effective in our context if it significantly cuts polyhedron P_2 .

In the short term, improvements are more likely to come from column generation. The goal is to devise efficient ways of pricing over a set of columns as close to cycle-free q -routes as possible. The s -cycle elimination approach we chose was reasonably effective, at least for $s \leq 4$. But other alternatives to further restrict the q -routes can also be tried. One could, for instance, forbid q -routes that visit vertices in a small set S more than once. This set could be dynamically chosen in order to eliminate q -routes with long cycles that are particularly harmful to bound quality.

In principle, several of the combinatorial relaxations of the CVRP—such as k -degree center trees, K -trees, or b -matchings—could also be used instead of (or even in addition to) q -routes. We think they are unlikely to have a noticeable impact, for two reasons. First, among the relaxations mentioned above, only q -routes take client demands and vehicle capacities into account. These “numerical” elements are precisely what makes the CVRP much harder in practice than a “pure” graph problem such as the TSP. It seems that known families of inequalities for the CVRP, mostly inspired by previous work on the TSP, cannot cope well with this numerical aspect. The inequalities implicitly given by P_2 can do better. The second reason for not using the alternatives above is that only q -routes lead to a pricing problem that is superpolynomial, but still practical. The polyhedra associated with polynomial pricing problems can be fully described and efficiently separated, which usually makes a pure BC (instead of BCP) a faster alternative. For instance, we can separate over the b -matching polyhedron in a very efficient way [21]. Separation over the q -route polyhedron, on the other hand, is possible only implicitly, with column generation.

Even if we leave column and cut generation as they are, we can still accelerate the algorithm through better integration. The task of designing, coding and fine-tuning a high performance branch-and-cut or branch-and-price procedure is complex, requiring experience to choose among several possible strategies, and, often, some computational tricks. BCP multiplies the possibilities and, therefore, the difficulties. We believe that there is still plenty of room for such improvements in our BCP.

⁶ A. Letchford proved that generalized large multistar inequalities do not cut the polyhedron induced by *cycle-free* q -routes (personal communication). Although this is not necessarily true when using q -routes without s -cycles, the heuristic separation in our BCP never found a violated GL multistar.

Acknowledgements. RF was supported by NSF grant DMI-0245609, having started this work while at Gapso Inc., Brazil. MPA was funded by research grant 300475/93-4 from CNPq. MR was supported by CAPES. EU was funded by scholarships from UFF and CNPq. RW was supported by NSF grant 11201878-123412. We thank Humberto Longo for his help during implementation and with experiments.

References

1. Achuthan, N., Caccetta, L., Hill, S.: Capacited vehicle routing problem: Some new cutting planes. *Asia-Pacific J. of Operational Research* **15** (1998) 109–123
2. Achuthan, N., Caccetta, L., Hill, S.: An improved branch-and-cut algorithm for the capacitated vehicle routing problem. *Transportation Science* **37** (2003) 153–169
3. Araque, J., Hall, L., Magnanti, T.: Capacitated trees, capacitated routing and associated polyhedra. Technical Report SOR-90-12, Princeton University (1990)
4. Araque, J., Kudva, G., Morin, T., Pekny, J.: A branch-and-cut algorithm for the vehicle routing problem. *Annals of Operations Research* **50** (1994) 37–59
5. Augerat, P.: Approche polyédrale du problème de tournées de véhicules. PhD thesis, Institut National Polytechnique de Grenoble (1995)
6. Augerat, P., Belenguer, J., Benavent, E., Corberán, A., Naddef, D., Rinaldi, G.: Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report 949-M, Université Joseph Fourier, Grenoble, France (1995)
7. Agarwal, Y., Mathur, K., Salkin, H.: A set-partitioning based exact algorithm for the vehicle routing problem. *Networks* **19** (1989) 731–739
8. Balinski, M., Quandt, R.: On an integer program for a delivery problem. *Operations Research* **12** (1964) 300–304
9. Blasum, U., Hochstättler, W.: Application of the branch and cut method to the vehicle routing problem. Technical Report ZPR2000-386, Zentrum für Angewandte Informatik Köln (2000)
10. Christofides, N., Eilon, S.: An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly* **20** (1969) 309–318
11. Christofides, N., Mingozzi, A., Toth, P.: Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming* **20** (1981) 255–282
12. Cornuéjols, G., Harache, F.: Polyhedral study of the capacitated vehicle routing problem. *Mathematical Programming* **60** (1993) 21–52
13. Dantzig, G., Ramser, R.: The truck dispatching problem. *Management Science* **6** (1959) 80–91
14. Fisher, M.: Optimal solution of vehicle routing problem using minimum k-trees. *Operations Research* **42** (1994) 626–642
15. Fukasawa, R., Poggi de Aragão, M., Porto, O., Uchoa, E.: Robust branch-and-cut-and-price for the capacitated minimum spanning tree problem. In: Proc. of the International Network Optimization Conference, Evry, France (2003) 231–236
16. Fukasawa, R., Reis, M., Poggi de Aragão, M., Uchoa, E.: Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Technical Report RPEP Vol.3 no.8, Universidade Federal Fluminense, Engenharia de Produção, Niterói, Brazil (2003)

17. Hadjiconstantinou, E., Christofides, N., Mingozzi, A.: A new exact algorithm from the vehicle routing problem based on q -paths and k -shortest paths relaxations. In Laporte, G., Gendreau, M., eds.: Freight Transportation. Number 61 in Annals of Operations Research. Baltzer Science Publishers (1995) 21–44
18. Irnich, S., Villeneuve, D.: The shortest path problem with resource constraints and k -cycle elimination for $k \geq 3$. Unpublished manuscript (2003)
19. Laporte, G., Norbert, Y.: A branch and bound algorithm for the capacitated vehicle routing problem. Operations Research Spektrum **5** (1983) 77–85
20. Letchford, A., Eglese, R., Lysgaard, J.: Multistars, partial multistars and the capacitated vehicle routing problem. Mathematical Programming **94** (2002) 21–40
21. Letchford, A., Reinelt, G., Theis, D.: A faster exact separation algorithm for blossom inequalities. This volume (2004)
22. Lysgaard, J.: CVRPSEP: A package of separation routines for the capacitated vehicle routing problem (2003) Available at www.asb.dk/~lys.
23. Lysgaard, J., Letchford, A., Eglese, R.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming (2003) (To appear).
24. Martinhon, C., Lucena, A., Maculan, N.: A relax and cut algorithm for the vehicle routing problem. European J. of Operational Research (2003) (To appear).
25. Miller, D.: A matching based exact algorithm for capacitated vehicle routing problems. ORSA J. on Computing **7** (1995) 1–9
26. Naddef, D., Rinaldi, G.: Branch-and-cut algorithms for the capacitated VRP. In Toth, P., Vigo, D., eds.: The Vehicle Routing Problem. SIAM (2002) 53–84
27. Pigatti, A.: Modelos e algoritmos para o problema de alocação generalizada e aplicações. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro, Brazil (2003)
28. Poggi de Aragão, M., Uchoa, E.: Integer program reformulation for robust branch-and-cut-and-price. In: Annals of Mathematical Programming in Rio, Búzios, Brazil (2003) 56–61
29. Ralphs, T.: Parallel branch and cut for capacitated vehicle routing. Parallel Computing **29** (2003) 607–629
30. Ralphs, T., Kopman, L., Pulleyblank, W., Jr., L.T.: On the capacitated vehicle routing problem. Mathematical Programming **94** (2003) 343–359
31. Toth, P., Vigo, D.: Models, relaxations and exact approaches for the capacitated vehicle routing problem. Discrete Applied Mathematics **123** (2002) 487–512
32. Toth, P., Vigo, D.: The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. SIAM (2002)
33. Wenger, K.: Generic Cut Generation Methods for Routing Problems. PhD thesis, Institute of Computer Science, University of Heidelberg (2003)
34. Werneck, R.F., Setubal, J.C.: Finding minimum congestion spanning trees. ACM J. of Experimental Algorithmics **5** (2000)

Metric Inequalities and the Network Loading Problem

Pasquale Avella¹, Sara Mattia², and Antonio Sassano²

¹ RCOST - Research Center on Software Technology
Università del Sannio
Viale Traiano
82100 Benevento - Italy
avella@unisannio.it

² Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Buonarroti, 12
00185 Roma - Italy
{mattia, sassano}@dis.uniroma1.it

Abstract. Given a simple graph $G(V, E)$ and a set of traffic demands between the nodes of G , the *Network Loading Problem* consists of installing minimum cost integer capacities on the edges of G allowing routing of the traffic demands.

In this paper we study the *Capacity Formulation* of the Network Loading Problem, introducing the new class of the Tight Metric Inequalities, that completely characterize the convex hull of the integer feasible solutions of the problem. We present separation algorithms for Tight Metric Inequalities and a cutting plane algorithm, reporting on computational experience.

1 Introduction

Let $G(V, E)$ be a graph without loops and multiple edges. A set of traffic demands have to be routed between the nodes of G . Let d_{ij} denote the demand between nodes i and j and let $D(V, H)$ be the (directed) *demand graph*, where $H = \{ij \in V \times V : d_{ij} > 0\}$ and $d : H \rightarrow \mathbb{R}_+$.

A capacity must be installed on the edges of G in integer multiples of a base unit. A set of capacities $\bar{X} = \{\bar{x}_{ij} : ij \in E\}$ is said *feasible* if all the demands can be routed simultaneously with the constraint that the total flow traversing $ij \in E$ does not exceed \bar{x}_{ij} .

Let c_{ij} be the cost of installing a unit capacity on the edge $ij \in E$ and let $c(\bar{X}) = \sum_{ij \in E} c_{ij} \bar{x}_{ij}$ denote the cost of \bar{X} . The *Network Loading Problem*, denoted by $NL(c, G, d)$, is the problem of determining a set of feasible capacities X minimizing $c(X)$. Let $\rho(c, G, d)$ denote the value of the optimal solution of $NL(c, G, d)$.

We refer to the *Asymmetric Network Loading Problem* if a directed graph $\vec{G}(V, A)$ is used.

Let $O = \{o \in V : \sum_{i \in V} d_{oi} > 0\}$ be the set of the *source nodes* of G . By associating a commodity with each source node $o \in O$, we can write the *Flow Formulation*, where f_{ij}^o denotes the amount of commodity o traversing the edge ij :

$$\min \sum_{ij \in E} c_{ij} x_{ij} \quad (1)$$

$$\sum_{j:ij \in E} f_{ij}^o - \sum_{j:ij \in E} f_{ji}^o = -d_i^o, \quad o \in O, i \in V \quad (1)$$

$$\sum_{o \in O} (f_{ij}^o + f_{ji}^o) \leq x_{ij}, \quad ij \in E \quad (2)$$

$$f_{ij}^o \geq 0, \quad o \in O, ij \in E \quad (3)$$

$$x_{ij} \in \mathbb{Z}^+, \quad ij \in E \quad (4)$$

Flow constraints (1) guarantee that the traffic demand d_i^o is routed from the source node o to each node $i \in V$. Capacity constraints (2) ensure that the total flow traversing ij does not exceed the installed capacity x_{ij} .

By projecting flow variables, the Network Loading Problem can be formulated in the space defined by the capacity variables x_{ij} .

Let $i, j \in V$ and let Π_{ij} denote the family of the paths between i and j on G . The function $\mu : E \rightarrow \mathbb{R}_+^{|E|}$ defines a *metric* on G if $\sum_{hk \in P_{ij}} \mu_{hk} - \mu_{ij} \geq 0$, for each $P_{ij} \in \Pi_{ij}$ and $ij \in E$ [31], and we refer to the set $Met(G) = \{\mu \in \mathbb{R}_+^{|E|} : \sum_{hk \in P_{ij}} \mu_{hk} - \mu_{ij} \geq 0, P_{ij} \in \Pi_{ij}, ij \in E\}$ as the *metric cone*. Let ℓ_j^i denote the shortest distance between nodes i and j according to the weights μ .

By using the feasibility condition for multicommodity flows introduced in [30][34][28], the *Capacity Formulation* for the Network Loading Problem is:

$$\begin{aligned} \min & \sum_{ij \in E} c_{ij} x_{ij} \\ \mu x & \geq \ell d, \quad \mu \in Met(G) \\ x & \in \mathbb{Z}_+^{|E|} \end{aligned} \quad (5)$$

where inequalities (5) are referred to as *Metric Inequalities*.

Let $\mu x \geq \ell d$ be any Metric Inequality. The Capacity Formulation can be strengthened by studying the Knapsack Polyhedron $KNAP(\mu) = \text{Conv} \{x \in \mathbb{Z}_+^{|E|} : \mu x \geq \ell d\}$. Valid inequalities for $KNAP(\mu)$ can be derived by multiplying both the sides by λ and then rounding-up the l.h.s. coefficients and the r.h.s..

Let $\mu \in \mathbb{Z}^{|E|}$. Inequalities of the form $\mu x \geq \lceil \ell d \rceil$, are referred to as *Rounded Metric Inequalities*. Let S, T be two disjoint subsets of V defining a partition and let $(S : T) = \{ij : i \in S, j \in T\}$. Special kinds of Rounded Metric Inequalities are the *Cut Inequalities* $x(S : T) \geq \lceil d(S : T) + d(T : S) \rceil$.

The Network Loading Problem has received a considerable amount of attention in the literature [3].

Magnanti, Mirchandani and Vachani [32][33] studied basic polyhedral properties of the problem and introduced three basic classes of valid inequalities: Cut Inequalities, 3-Partition Inequalities and Arc Residual Capacity Inequalities. Atamturk and Rajan [2] presented a linear-time separation algorithm for Arc Residual Capacity Inequalities.

Barahona [4] presented a cutting plane algorithm based on Cut Inequalities, to provide lower bounds for the Network Loading Problem. The separation problem for Cut Inequalities is formulated as a Max-Cut problem. Barahona also introduced Spanning Tree Inequalities to enforce connectivity of feasible solutions. The solution of the relaxed formulation is used to set lower bounds for the capacities and then a Branch-and-Bound based on the Flow Formulation is run to provide an upper bound.

Dahl and Stoer [21] studied a Network Loading Problem with survivability requirements. They present a cutting plane algorithm where violated Metric Inequalities are generated by solving a LP by a column generation approach.

Bienstock and Günlük [12] investigated the polyhedral structure of the Flow Formulation for the asymmetric version of the problem and developed a cutting plane algorithm. They derived the Flow Cutset Inequalities as a class of Mixed-Integer Rounding Inequalities. Chopra, Gilboa and Sastry [16] generalized the Flow Cutset Inequalities introduced in [12].

Bienstock, Chopra, Günlük and Tsai [10] compared cutting plane algorithms based, respectively, on the Flow and on the Capacity Formulation. Günlük [25] introduced new families of Mixed-Integer Rounding Inequalities.

Atamturk [1] studied the polyhedral properties of the Flow Formulation for the special Network Loading Problem over a "cutset", defined, given a nonempty partition (U_1, U_2) of V , as the set of arcs directed from U_1 to U_2 and vice-versa.

Solving the LP-relaxation of the Flow Formulation for large instances, requires a huge amount of memory. Crainic, Frangioni and Gendron [17][18] solved the lagrangian relaxation of the Flow Formulation by the Bundle Method. Bienstock [6][7][8] has developed a potential reduction approach to solve large-scale block structured linear programs. The algorithm turned out to be very effective to tackle large-scale Network Loading instances.

Holmberg and Yuan [27] proposed a Branch-and-Bound based on a lagrangian relaxation of the Flow Formulation for the case where $x_{ij} \in \{0, 1\}$, $ij \in E$. Local search heuristics have been investigated in [20] [19].

In this paper we study the Capacity Formulation of $NL(c, G, d)$, introducing the new class of the Tight Metric Inequalities, that completely characterize the convex hull of the integer feasible solutions of the problem. We present separation algorithms for Tight Metric Inequalities and a cutting plane algorithm, reporting on computational experience.

The remainder of the paper is organized as follows. In section 2 we introduce the Tight Metric Inequalities. In section 3 we address several separation algorithms for Metric and Tight Metric Inequalities. In section 4 we outline

the cutting plane algorithm. Finally in section 5 we report on computational experience with two sets of test instances.

2 Tight Metric Inequalities

Let $P(G, d) = \text{conv} \{x \in \mathbb{Z}_+^{|E|} : x \text{ is feasible}\}$ be the *Network Loading Polyhedron*. In this section we introduce the family of the Tight Metric Inequalities and we prove that they completely characterize $P(G, d)$.

We first show that the left-hand-side of any non-redundant valid inequality defines a metric.

Theorem 1. *Let $ax \geq b$ be any valid inequality for $P(G, d)$. There exists $\mu \in \text{Met}(G)$ such that:*

- i) $\mu x \geq b$ is valid
- ii) $\mu_{ij} \leq a_{ij}, \quad ij \in E$

Proof. Let i and j be two distinct nodes of V and let $a(P_{ij}^{\min})$ denote the length of the shortest path between i and j according to the weights a . Let $\mu_{ij} = a(P_{ij}^{\min}) \leq a_{ij}, ij \in E$. It follows that $\mu \in \text{Met}(G)$ and the inequality $\mu x \geq b$ dominates $ax \geq b$.

Suppose that $\mu x \geq b$ is not valid and let \bar{x} be a feasible solution with the property that $\mu \bar{x} < b$. Without loss of generality, we can assume that there is only one edge hk with $\mu_{hk} = a(P_{hk}^{\min}) < a_{hk}$.

We can build a new feasible solution \hat{x} , defined as follows:

$$\hat{x}_{ij} = \begin{cases} \bar{x}_{ij} & ij \neq hk, \quad ij \notin P_{hk}^{\min} \\ \bar{x}_{ij} + \bar{x}_{hk} & ij \in P_{hk}^{\min} \\ 0 & ij = hk \end{cases},$$

with the property that $a\hat{x} = \mu\bar{x} < b$. It follows that \hat{x} is feasible and $ax \geq b$ is violated, contradiction.

Let $\mu \in \text{Met}(G)$ and let $\rho(\mu, G, d)$ be the optimal solution of the Network Loading Problem where the vector μ replaces c in the objective function, i.e. $\rho(\mu, G, d) = \min\{\mu x : x \in \mathbb{Z}_+^{|E|}, x \text{ is feasible}\}$. Any valid inequality $\mu x \geq b$ for $P(G, d)$ is dominated by $\mu x \geq \rho(\mu, G, d)$. We refer to any inequality of the form $\mu x \geq \rho(\mu, G, d)$ as a *Tight Metric Inequality*.

Example 1. Let $G(V, E)$ be a complete graph on 3 nodes, $V = \{1, 2, 3\}$, and let $d_{12} = d_{13} = d_{23} = 1.2$. From the integer metric $\mu_{12} = \mu_{13} = \mu_{23} = 1$ we get the Metric Inequality $x_{12} + x_{13} + x_{23} \geq 3.6$, that can be strengthened to the Rounded Metric Inequality $x_{12} + x_{13} + x_{23} \geq 4$. But it can be easily checked that $\rho(\mu, G, d) = 5$ and the corresponding Tight Metric Inequality is $x_{12} + x_{13} + x_{23} \geq 5$.

The following result emphasizes the crucial role of the Tight Metric Inequalities.

Corollary 1. *The Tight Metric Inequalities $\mu x \geq \rho(\mu, G, d)$, $\mu \in \text{Met}(G)$, completely describe $P(G, d)$.*

A metric $\mu \in \text{Met}(G)$ is *extreme* if μ is an extreme ray of the metric cone $\text{Met}(G)$, i.e. if μ cannot be derived as a conic combination of two other distinct metrics. The following property holds if $\mu \in \mathbb{Z}^{|E|}$ is extreme.

Theorem 2. *If $\mu \in \mathbb{Z}^{|E|}$ is an extreme ray of $\text{Met}(G)$, then $\rho(\mu, G, d) = \lceil \ell d \rceil$.*

Proof. Let A be a $m \times n$ matrix and let $P_I = \text{conv}\{x \in \mathbb{Z}^n : Ax \geq b\}$.

Let $u \in \mathbb{R}_+^m$ be a vector of multipliers. Starting from the linear system $Ax \geq b$, the Chvátal-Gomory procedure produces a valid inequality for P_I of the type $\lceil uA \rceil x \geq \lceil ub \rceil$ and adds it to $Ax \geq b$. It is a well known result [36] that any valid inequality for P_I can be derived by applying the Chvátal-Gomory procedure a finite number of times.

Now let us turn to our problem, where $Ax \geq b$ is a set of Metric Inequalities. Let $\gamma = \lceil uA \rceil$, $\delta = \lceil ub \rceil$ and let $\gamma x \geq \delta$ be the inequality derived by the Chvátal-Gomory procedure. It can be easily proved that $\gamma \in \text{Met}(G)$. It follows that at any stage of the procedure the system A includes only inequalities of the form $\gamma x \geq b$, with $\gamma \in \text{Met}(G)$.

Let μ an extreme ray of $\text{Met}(G)$ and let $\mu x \geq k$ be valid for $P(G, d)$. Since μ cannot be obtained as a conic combination of two or more other distinct metrics, it follows that $\mu x \geq k$ can be derived applying the Chvátal-Gomory procedure with a vector of multipliers u where the multiplier associated with the $\mu x \geq \ell d$ itself is 1 and the multipliers associated with the other inequalities are null and then $k = \lceil \ell d \rceil$.

Let S and T be a partition of V and let $(S : T) = \{ij \in E : i \in S, j \in T\}$ be a cut on G . Deza and Laurent [22] proved that the metrics defined by cuts, i.e. $\mu_{ij} = 1$ for $ij \in (S : T)$ and $\mu_{ij} = 0$ for $ij \in A \setminus (S : T)$ are the only extreme metrics with $\mu \in \mathbb{Z}^{|E|}$. It follows that Cut Inequalities $x(S : T) \geq \lceil d(S : T) + d(T : S) \rceil$ are Tight Metric Inequalities.

3 Separation Algorithms

Separation of Tight Metric Inequalities is a difficult task, as even computing the r.h.s $\rho(\mu, G, d)$ for a given inequality $\mu x \geq b$ is NP-hard. We address the separation problem by a *two-stage heuristic*: *i*) we look for a violated inequality $ax \geq b$ and then *ii*) we turn it into the $\mu x \geq b$, and apply a shrinking procedure to compute $\rho(\mu, G, d)$ to get a Tight Metric Inequality of the form $\mu x \geq \rho(\mu, G, d)$.

The remainder of this section is organized as follows. In subsection 3.1 we discuss separation of Metric Inequalities, in subsection 3.2 we address the problem of finding violated Metric Inequalities with minimum support. In 3.4 and 3.5 we describe, respectively, separation of Metric Inequalities with $\mu \in \{0, 1\}^{|E|}$ and separation of Mod- k Cuts.

In section 3.6 we describe the shrinking procedure that, for a given metric μ , allows computing $\rho(\mu, G, d)$ on a reduced size graph.

3.1 Separation of Metric Inequalities

It is well known that the separation problem for Metric Inequalities amounts to checking whether there exists a feasible multicommodity flow with respect to a given capacity vector $\bar{x} \in \mathbb{R}_+^{|E|}$. The feasibility of a multicommodity flow problem can be tested by solving the following Linear Programming (auxiliary) problem $Sep1(G, d)$:

$$\max \alpha \quad (6)$$

$$\sum_{j:ij \in E} f_{ij}^o - \sum_{j:ji \in E} f_{ji}^o = -d_i^o \alpha, \quad i \in V, o \in O$$

$$\begin{aligned} \sum_{o \in O} (f_{ij}^o + f_{ji}^o) &\leq \bar{x}_{ij}, \quad ij \in E \\ f_{ij}^o &\geq 0, \quad o \in O, ij \in E \end{aligned} \quad (7)$$

If the above problem admits an optimal solution of value $\hat{\alpha} \geq 1$ then the multicommodity flow is feasible and no Metric Inequality is violated. If, conversely, the optimal solution has a value $\hat{\alpha} < 1$ then the multicommodity flow problem is infeasible and the optimal dual solution of the auxiliary problem provides a violated Metric Inequality. The above auxiliary problem is the well known Maximum Concurrent Flow Problem (see Günlük [26], Bienstock and Raskina [13] and Fleischer [24]). Its dual has the following form:

$$\begin{aligned} \min \sum_{ij \in E} \bar{x}_{ij} \mu_{ij} \\ (f_{ij}^o) \quad \ell_i^o - \ell_j^o + \mu_{ij} \geq 0, \quad o \in O, ij \in E \\ (f_{ji}^o) \quad \ell_j^o - \ell_i^o + \mu_{ij} \geq 0, \quad o \in O, ij \in E \\ (\alpha) \quad \sum_{o \in O} \sum_{i \in V} d_i^o \ell_i^o = 1 \\ \mu_{ij} \geq 0, \quad ij \in E \end{aligned} \quad (8)$$

Denoted by $(\hat{\mu}, \hat{\ell})$ its optimal solution and assuming that $\hat{\alpha} < 1$, we have that $\bar{x}\hat{\mu} < 1$ and, consequently, that $\bar{x}\hat{\mu} < \hat{\ell}d$. This implies that $\hat{\mu}x \geq \hat{\ell}d$ is the sought for a maximally violated Metric Inequality.

The following result proves that $\hat{\mu} \in Met(G)$.

Lemma 1. *Let $(\hat{\mu}, \hat{\ell})$ be the optimal solution of the separation oracle $Sep1(G, d)$. Then $\hat{\mu} \in Met(G)$.*

Proof. For any $hk \in E$, let $\mu(P_{hk}^{min})$ denote the length of the shortest path between h and k according to the weights μ . Suppose that $\hat{\mu} \notin Met(G)$. Then there exist $ij \in E$ such that $\hat{\mu}_{ij} > \hat{\mu}(P_{ij})$. But since ℓ_i^o is the length of the shortest path from o to i , we can decrease $\hat{\mu}_{ij}$ while keeping triangle inequalities satisfied. We get a better feasible solution and consequently $(\hat{\mu}, \hat{\ell})$ is not optimal, contradiction.

3.2 Separation of Strong Metric Inequalities

Estimating the quality of cutting planes is a major issue in Computational Integer Programming as, in general, violation cannot be considered as a good cut selection criterion. Bienstock and Bley [9] observe that looking for minimal support violated Cut Inequalities may lead to more effective cutting planes and suggest to perturb the fractional solution by a tiny value to get minimal support violated cutting planes.

In what follows we generalize such considerations by measuring the *depth* of a Metric Inequality as the ratio between violation and the size of the support of the inequality, defined as $\sum_{ij \in E} \mu_{ij}$. We refer to Metric Inequalities minimizing the ratio:

$$\frac{\sum_{ij \in E} \bar{x}_{ij} \mu_{ij} - \sum_{o \in O} \sum_{i \in V} d_i^o \ell_i^o}{\sum_{ij \in E} \mu_{ij}} \quad (9)$$

as *Strong Metric Inequalities*. The separation oracle for Strong Metric Inequalities is:

$$\begin{aligned} \min & \frac{\sum_{ij \in E} \bar{x}_{ij} \mu_{ij} - \sum_{o \in O} \sum_{i \in V} d_i^o \ell_i^o}{\sum_{ij \in E} \mu_{ij}} \\ & \ell_i^o - \ell_j^o + \mu_{ij} \geq 0, \quad o \in O, ij \in E \\ & \ell_j^o - \ell_i^o + \mu_{ij} \geq 0, \quad o \in O, ij \in E \\ & \mu_{ij} \geq 0, \quad ij \in E \end{aligned} \quad (10)$$

We prove that the separation oracle for Strong Metric Inequalities reduces to the *Capacity Reduction Problem Sep2(G, d)*:

$$\begin{aligned} & \max \beta \\ & \sum_{j:ij \in E} f_{ij}^o - \sum_{j:ji \in E} f_{ji}^o = -d_i^o, \quad o \in O, i \in V \\ & \sum_{o \in O} (f_{ij}^o + f_{ji}^o) \leq \bar{x}_{ij} - \beta, \quad ij \in E \\ & f_{ij}^o \geq 0, \quad o \in O, ij \in E \end{aligned}$$

If the above problem admits an optimal solution of value $\hat{\beta} \geq 0$ then the multicommodity flow is feasible and no Metric Inequality is violated. If, conversely, the optimal solution has a value $\hat{\beta} < 0$ then, as in the previous case, the optimal dual solution will provide us with a violated constraint. The dual of *Sep2(G, d)*, say *Sep2dual(G, d)* is:

$$\begin{aligned}
& \min \sum_{ij \in E} \bar{x}_{ij} \mu_{ij} - \sum_{o \in O} \sum_{i \in V} \ell_i^o d_i^o \\
& \ell_i^o - \ell_j^o + \mu_{ij} \geq 0, \quad o \in O, ij \in E \\
& \ell_j^o - \ell_i^o + \mu_{ij} \geq 0, \quad o \in O, ij \in E \\
& \sum_{ij \in E} \mu_{ij} = 1 \\
& \mu_{ij} \geq 0, \quad ij \in E
\end{aligned} \tag{11}$$

Denoted by $(\hat{\mu}, \hat{l})$ its optimal solution and assuming that $\hat{\beta} < 0$, we have that $\bar{x}\hat{\mu} - d\hat{l} < 0$. This implies that $\hat{\mu}x \geq \hat{l}d$ is the sought for maximally violated Metric Inequality.

Theorem 3. *The separation oracle (11), provides a Metric Inequality minimizing the ratio (9).*

Proof. Consider the fractional linear program (10). By letting $1/\sum_{ij \in E} \mu_{ij} = t$, we can write the equivalent problem:

$$\begin{aligned}
& \min \sum_{ij \in E} t\bar{x}_{ij} \mu_{ij} - \sum_{o \in O} \sum_{i \in V} t d_i^o \ell_i^o \\
& t\ell_i^o - t\ell_j^o + t\mu_{ij} \geq 0, \quad o \in O, ij \in E \\
& t\ell_j^o - t\ell_i^o + t\mu_{ij} \geq 0, \quad o \in O, ij \in E \\
& t \sum_{ij \in E} \mu_{ij} = 1 \\
& \mu_{ij} \geq 0, \quad ij \in E
\end{aligned}$$

Then, by letting $t\mu_{ij} = w_{ij}$ and $t\ell_j^o = v_j^o$, we get:

$$\begin{aligned}
& \min \sum_{ij \in E} \bar{x}_{ij} w_{ij} - \sum_{o \in O} \sum_{i \in V} v_i^o d_i^o \\
& v_i^o - v_j^o + w_{ij} \geq 0, \quad o \in O, ij \in E \\
& v_j^o - v_i^o + w_{ij} \geq 0, \quad o \in O, ij \in E \\
& \sum_{ij \in E} w_{ij} = 1 \\
& w_{ij} \geq 0, \quad ij \in E \\
& t \geq 0
\end{aligned}$$

So we have obtained $Sep2dual(G, d)$ by linearizing the fractional linear programming problem (10).

In table 1 we show for a set of test instances that separation of Strong Metric Inequalities yields a strong reduction in the number of violated cutting planes used to provide a lower bound.

Name	LB	# Metric	#Strong
Sun.tr1	2753.39	402	135
Sun.tr2	2791.53	480	175
Sun.tr3	3067.96	473	191
Sun.tr4	2829.51	456	163
Sun.tr5	2391.3	412	188
Sun.tr6	2997.64	450	175

Table 1. Comparison of separation oracles $Sep1(G, d)$ and $Sep2(G, d)$.

3.3 A Lifting Procedure for Metric Inequalities

Working on the fractional support $S = \{ij \in E : \bar{x}_{ij} > 0\}$ is crucial to make separation algorithms more efficient.

Let $G_S(V, S)$ be the fractional support graph. Here we present a lifting procedure that extends a Metric Inequality defined on G_S to a globally valid Metric Inequality defined on G .

Let $(\hat{\mu}_S, \hat{\ell})$ be the optimal dual variables of the separation problem. The vector $\hat{\mu}_S$ can be extended to the metric μ defined on G by letting:

$$\mu_{ij} = \max\{|\ell_j^o - \ell_i^o|, o \in O\}, \quad ij \in E \setminus S$$

The lifting procedure is exact for $Sep1(G, d)$, i.e. it guarantees that the Metric Inequality $\mu_S x_S \geq \ell d$ found on G_S and the globally valid Metric Inequality $\mu_S x_S \geq \ell d$, obtained by the lifting procedure, have the same amount of violation. For the Strong Metric Inequalities the lifting procedure ensures that a violated Metric Inequality exists on G if and only if a violated Metric Inequality exists on G_S , but there is no guarantee that $\mu x \geq \ell d$ has the same ratio (9) as $\mu_S x_S \geq \ell d$.

3.4 Separation of $\{0, 1\}$ -Rounded Metric Inequalities

For some classes of instances it turned out to be effective formulating the separation problem of $\{0, 1\}$ -Rounded Metric Inequalities as a Mixed-Integer Linear Programming problem:

$$\begin{aligned} \min \quad & \sum_{ij \in E} \bar{x}_{ij} \mu_{ij} - z \\ \text{s.t.} \quad & \ell_i^o - \ell_j^o + \mu_{ij} \geq 0, \quad o \in O, ij \in E \\ & \ell_j^o - \ell_i^o + \mu_{ij} \geq 0, \quad o \in O, ij \in E \\ & z < \sum_{o \in O} \sum_{i \in V} d_i^o \ell_i^o + 1 \\ & \mu_{ij} \in \{0, 1\}, \quad ij \in E \\ & z \in \mathbb{Z}_+ \end{aligned} \tag{12}$$

Constraint (12) enforces $z = \lceil \ell d \rceil$. If the optimal objective value is negative, we have found a violated $\{0, 1\}$ -Rounded Metric Inequality $\mu x \geq z$.

We observe that $\{0, 1\}$ -Rounded Metric Inequalities include Cut Inequalities, which are Tight Metric Inequalities, as observed in Section 2.

3.5 Mod- k Cuts

Given the integer polyhedron $P_I = \text{conv}\{x \in \mathbb{Z}^n : Ax \geq b\}$, where $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$, a Chvátal-Gomory cut is a valid inequality for P_I of the type $\lambda Ax \geq \lceil \lambda b \rceil$ for some $\lambda \in \mathbb{R}_+^m$ such that $\lambda A \in \mathbb{Z}^n$.

A Chvátal-Gomory Cut is a *mod- k cut* if $\lambda_i \in \{0, 1/k, \dots, (k-1)/k\}$ for some integer $k \geq 2$ ([14], [15], [29]). Every non dominated Chvátal-Gomory Cut is a Mod- k Cut for some k . For any given $x^* \in P$, the vector $s^* = Ax^* - b$ is called slack vector. A cut is said to be *totally tight* at x^* if $s_j^* \lambda_j = 0$ holds for $j = 1, \dots, m$.

The separation of a violated Chvátal-Gomory Cut was shown to be strongly *NP-hard* [23]. The same happens for Mod- k cuts [14], but to find a totally tight Mod- k Cut whose violation is maximal, or prove that none exists, can be done in polynomial time, provided that the prime factorization of k is known, by solving a congruence system in $GF(k)$, as described in [15].

If inequalities in the matrix A are Metric Inequalities, that is they have left-hand-side coefficients that define a metric, any inequality obtained by combining them is still a Metric Inequality, with a possibly larger r.h.s. It follows that Chvátal-Gomory Cuts and Mod- k Cuts can be seen as an intermediate step towards Tight Metric Inequalities. They also include, as special case, some Tight Metric Inequalities, like the Three-Partition Inequalities [32].

3.6 Computing $\rho(\mu, G, d)$: a Shrinking Procedure

Here we address step *ii*) of the separation heuristic for Tight Metric Inequalities. Any valid inequality $\mu x \geq b$, generated as described in sections 3.1-3.5, can be tightened by replacing the r.h.s. b with $\rho(\mu, G, d)$.

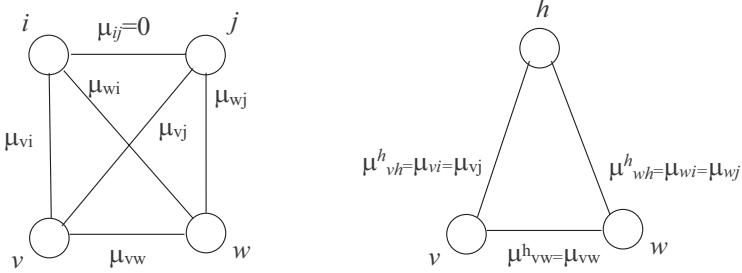
To compute $\rho(\mu, G, d)$ efficiently, we introduce a shrinking procedure that builds a reduced size problem $NL(\mu_h, G_h, d_h)$ with the property that $\rho(\mu_h, G_h, d_h) = \rho(\mu, G, d)$.

Let $ij \in E$ such that $\mu_{ij} = 0$ and let $G^h(V^h, E^h)$ be the graph obtained by shrinking the nodes i and j into the supernode h , thus $V^h = V \setminus \{i, j\} \cup \{h\}$. The edge set E^h is defined by merging all the pair of edges $vi, vj \in E$ into $vh \in E^h$, for each $v \in V \setminus \{i, j\}$.

Let μ^h denote the mapping of μ onto G^h . We observe that, due to the triangle inequalities $\mu_{vj} \leq \mu_{vi} + \mu_{vj}$, $\mu_{vi} \leq \mu_{vj} + \mu_{uj}$ and $\mu_{ij} = 0$, we get $\mu_{vi} = \mu_{vj}$. For any $v \in V^h \setminus \{h\}$ we set $\mu_{vh}^h = \mu_{vi} = \mu_{vj}$ and $\mu_{uv}^h = \mu_{uv}$, for each edge of the set $\{uv \in E : u, v \in V \setminus \{i, j\}\}$. The following result is due to Lomonosov [30]:

Theorem 4. *Let μ be a metric. The vector μ^h is a metric if $\mu_{ij} = 0$.*

Accordingly, let D^h be graph obtained by shrinking the nodes i and j in the demand graph D and let d_h be the mapping of d onto D^h . We have $d_{uv}^h = d_{uv}$ for any $u, v \in V^h \setminus \{h\}$, $d_{vh}^h = d_{vi} + d_{vj}$ and $d_{hv}^h = d_{iv} + d_{jv}$.

**Fig. 1.** The shrinking procedure.

Let $\rho(\mu^h, G^h, d^h)$ denote the optimal solution of the reduced problem $NL(\mu^h, G^h, d^h)$. The following property holds

Theorem 5. Let μ be a metric on G . Let $ij \in E$ such that $\mu_{ij} = 0$ and let μ^h, g^h, d^h be defined as above. Then $\rho(\mu, G, d) = \rho(\mu^h, G^h, d^h)$.

Proof. Suppose that $\rho(\mu^h, G^h, d^h) < \rho(\mu, G, d)$ and let x^h be the optimal solution corresponding to $\rho(\mu^h, G^h, d^h)$. We can construct a feasible solution x for the original problem $NL(\mu, G, d)$ as follows. Let $x_{ij} = \lceil \sum_{uv \in H} d_{uv} \rceil$, let $x_{uv} = x_{uv}^h, uv \in E$ and let $x_{vi} + x_{vj} = x_{vh}^h$. We know that $\mu x = \mu^h x^h = \rho(\mu^h, G^h, d^h) < \rho(\mu, G, d)$, contradiction because $\rho(\mu, G, d)$ is the minimum.

Now suppose that $\rho(\mu^h, G^h, d^h) > \rho(\mu, G, d)$ and let x be the optimal solution of $\rho(\mu, G, d)$. Let x^h be the mapping of x onto G^h . Then $\mu^h x^h = \mu x < \rho(\mu, G, d)$, contradiction because $\rho(\mu^h, G^h, d^h)$ is the minimum.

Applying the shrinking procedure iteratively we build a reduced size graph. If the size of the reduced graph is less than a given threshold (≤ 8 nodes in our experience), we can efficiently compute $\rho(\mu, G, d)$.

4 The Cutting Plane Algorithm

A cutting plane algorithm has been implemented to validate the effectiveness of the proposed separation procedures.

The skeleton of the algorithm is Cplex 8.1 that provides search tree and cut pool management functionalities. We adopt the default branching strategy of Cplex 8.1. Variable selection is performed by strong branching.

4.1 Preprocessing

A simple operation is used to delete redundant variables: let $c(P_{ij})$ be the length of the shortest path between i and j according to the objective function costs. We delete the edge ij if $c_{ij} > c(P_{ij})$.

Let us consider the optimal solution of the LP-relaxation, whose value will be denoted by LB . For any nonbasic variable we can fix bounds by *reduced cost fixing*. Let LB be the value of the LP-relaxation and let UB be the value of the best feasible solution found so far. Let r_{ij} be the reduced cost of variable x_{ij} . For each nonbasic variable x_{ij} we compute $\delta = \lfloor \frac{UB-LB}{|r_{ij}|} \rfloor$ and we fix $u_{ij} = \min(u_{ij}, l_{ij} + \delta)$ and $l_{ij} = \max(l_{ij}, u_{ij} - \delta)$.

4.2 Initial LP-relaxation

The initial LP-relaxation includes the Degree Inequalities $x(\delta(i)) \geq \lceil d(\delta(i)) \rceil$, $i \in V$.

4.3 Separation Strategy

At the root node of the Branch-and-Cut tree we run separation routines in the following priority order: Rounded Strong Metric Inequalities, Mod- k cuts. Once a violated cut has been found, we apply the shrinking procedure of section 3.6 to get a Tight Metric Inequality.

At the deeper nodes we look only for violated Rounded Strong Metric Inequalities to guarantee feasibility.

Given the Metric Inequality $\mu x \geq \ell d$, we transform it into the Rounded Metric Inequality $\lceil \lambda \mu \rceil x \geq \lceil \lambda \ell d \rceil$ by letting $\lambda = \min\{\mu_{ij} : ij \in E\}$, as suggested in [10].

4.4 Computing Upper Bounds

Let $\bar{X} = \{\bar{x}_{ij} : ij \in E\}$ be the optimal (fractional) solution of the LP-relaxation of the Capacity Formulation, i.e. X satisfies all the Metric Inequalities. A *trivial upper bound* can be computed by rounding-up the x_{ij} , for $ij \in E$.

The trivial upper bound can be improved by a LP-based heuristic. We impose the bounds $\lfloor \bar{x}_{ij} \rfloor$ and $\lceil \bar{x}_{ij} \rceil$, for each $ij \in E$ and we solve a *restricted problem* with the new bounds, yielding a minimum cost round-up of the \bar{x}_{ij} that preserves feasibility.

Alternatively, the trivial upper bound can be improved by a *re-routing heuristic* that attempts to round-down some of the fractional \bar{x}_{ij} .

For each $hk \in E$, let $s_{hk} = \lceil \bar{x}_{hk} \rceil - \bar{x}_{hk}$ be the *slack* of hk .

For any candidate edge $ij \in E$, the heuristic attempts to re-route the amount of flow $f_{ij} = \bar{x}_{ij} - \lfloor \bar{x}_{ij} \rfloor$ along a path P_{ij} from i to j formed by edges with $s_{hk} \geq f_{ij}$, $hk \in P_{ij}$. If such a path exists, we set $\bar{x}_{ij} = \lfloor \bar{x}_{ij} \rfloor$, yielding a new feasible fractional solution that provides a better upper bound. We found useful to sort the candidate edges ij in non-increasing order according to the value $c_{ij}(\lceil \hat{x}_{ij} \rceil - \hat{x}_{ij})$.

At every node of the Branch-and-Cut tree we run a *plunging heuristic*. Let \bar{X} be a fractional solution satisfying Metric Inequalities. If \bar{X} is near-integer, i.e. if $\bar{x}_{ij} \leq \lfloor \bar{x}_{ij} \rfloor + \varepsilon \vee \bar{x}_{ij} \geq \lceil \bar{x}_{ij} \rceil - \varepsilon$, for $ij \in E$, then we plunge x_{ij} to the closest bound to get an integer solution Y and if Y is feasible we can update the current upper bound.

5 Computational Results

In this section we report on the computational experience with the cutting plane algorithm. We ran the experiments on a Pentium IV 1.7 Ghz Personal Computer, with 512 Mb RAM, using Cplex 8.1 as *LP* solver and as Branch-and-Cut framework.

The test bed is organized into two sets. The first set includes the well-known Norwegian instances, whose asymmetric version is also tested. The second set includes instances derived as backbone networks from two real-life instances introduced by Barahona [4].

The format of the tables is the following. *Name* is the name of the instance, LB_{Flow} is the optimal value of the LP-relaxation of the Flow Formulation, LB_{Cuts} is the lower bound after the addition of cutting planes, BLB is the best lower bound yielded by the Branch-and-Cut, UB_{Heu} is the upper bound produced either by the LP-based or by the re-routing heuristic, BUB is best upper bound produced by the Branch-and-Cut (we use boldface to denote the upper bounds proved to be optimal), *Gap* is the final gap, computed as $100 * (BUB - BLB) / BUB$, $\#RM$ and $\#Mod-k$ are respectively the total number of Rounded Strong Metric Inequalities and the total number of Mod- k cuts added to the formulation. Finally, columns *Nodes* and *Time* report respectively on the number of nodes of the Branch-and-Cut tree and on total CPU time (secs.) spent in computation.

5.1 Norwegian Instances

Sun networks, with 27 nodes and 51 edges, were introduced in [21] and are known in the literature as the *Norwegian networks*. For each network we have tested the two different traffic patterns proposed in [10]: *Sun.tr*, 67 demand pairs with magnitude in the interval [0.1, 1.2] and *Sun.dense*, 702 demand pairs with magnitude in the interval [1, 2]. For the instances in this class the LP-based upper bound heuristic, whose results are reported in the column UB_{Heu} , turned out to be very effective, both in the symmetric and in the asymmetric case.

<i>Name</i>	LB_{Flow}	LB_{Cuts}	BLB	UB_{Heu}	BUB	<i>Gap</i> %	$\#RM$	$\#Mod-k$	<i>Nodes</i>	<i>Time</i>
<i>SUN.tr1</i>	2738.78	2822.52	2825.44	2830.74	2825.44	0.00	138	34	31	17
<i>SUN.tr2</i>	2773.72	2862.19	2869.92	3150.98	2865.30	0.00	171	31	19	78
<i>SUN.tr3</i>	3038.38	3139.19	3153.13	3153.67	3153.13	0.00	160	42	87	401
<i>SUN.tr4</i>	2824.11	2891.19	2894.41	2895.9	2894.41	0.00	121	32	22	632
<i>SUN.tr5</i>	2375.11	2460.47	2472.10	2473.34	2472.10	0.00	171	142	91	58
<i>SUN.tr6</i>	2979.25	3075.32	3086.67	3090.34	3086.67	0.00	206	23	562	168

Table 2. Computational results for the *SUN.tr* instances

Name	LB_{Flow}	LB_{Cuts}	BLB	UB_{Heu}	BUB	Gap%	#RM	#Mod- k	Nodes	Time
$SUN.dense1$	29594	29650.9	29651.3	29651.3	29651.3	0.00	188	33	5	91
$SUN.dense2$	29583.9	29646.6	29653.6	29654.2	29653.6	0.00	282	30	663	412
$SUN.dense3$	98575.5	98639.8	3153.1	98640.5	98640.5	0.00	186	30	10	592
$SUN.dense4$	98347.9	98406.0	2894.4	98414.4	98410	0.00	209	41	130	563
$SUN.dense5$	59115	59163.2	59166.8	59167.4	59166.8	0.00	217	16	36	508
$SUN.dense6$	58923.4	58985.6	58987.8	58990.2	58987.8	0.00	183	30	25	295

Table 3. Computational results for the $SUN.dense$ instances

Asymmetric Norwegian Instances Asymmetric instances, denoted by $\overrightarrow{Sun.tr}$ and $\overrightarrow{Sun.dense}$, are derived, respectively, from $Sun.tr$ and $Sun.dense$ by replacing every edge $ij \in E$ with two directed arcs ij, ji with costs $c_{ij} = c_{ji}$. Previous results for these instances are reported in [10] and, as far as our knowledge, none of these instances, with the only exception of $\overrightarrow{Sun.tr4}$, have been solved to exact optimality before.

For the $\overrightarrow{Sun.tr}$ instances, Mod- k cuts turned out to be ineffective and were replaced with the $\{0, 1\}$ -Rounded Metric Inequalities, whose separation is performed using the MIP problem of section 3.4. Let #01RM denote the number of $\{0, 1\}$ -Rounded Metric Inequalities added in the search tree.

In Table 4 we report computational results for the $\overrightarrow{Sun.tr}$ instances, that were solved to optimality.

Name	LB_{Flow}	LB_{Cuts}	BLB	UB_{Heu}	BUB	Gap%	#RM	#Mod- k	Nodes	Time
$Sun.tr1$	2738.78	2962.42	2962.42	2979.49	2962.42	0.00	3029	161	5914	4523
$Sun.tr2$	2773.72	2976.24	2976.24	2980.71	2976.24	0.00	1684	94	1928	1733
$Sun.tr3$	3038.38	3242.78	3242.78	3257.22	3242.78	0.00	1406	73	426	1880
$Sun.tr4$	2824.11	2978.90	2978.90	2978.9	2978.90	0.00	506	132	46	7103
$Sun.tr5$	2375.11	2595.00	2595.00	2609.87	2595.00	0.00	2033	113	1591	8947
$Sun.tr6$	2979.25	3196.96	3196.96	3196.96	3196.96	0.00	8539	108	4487	20464

Table 4. Computational results for the $\overrightarrow{Sun.tr}$ instances

Table 5 reports on results for $\overrightarrow{Sun.dense}$ instances. Due to the larger number of demand pairs and of source nodes, separation of Strong Metric Inequalities becomes more difficult. For such instances we were not able to prove exact optimality, but we yielded small gaps and improved the results reported in [10].

5.2 US and German Instances

In [4] Barahona introduced two real-life instances defined on complete graphs, that were kindly made available to us by the author: *Bar41*, a US instance with

Name	LB_{Flow}	LB_{Cuts}	BLB	UB_{Heu}	BUB	Gap%	#RM	#Mod- k	Nodes	Time
$\overrightarrow{Sun.dense}1$	29594.0	29721.8	29732.2	30641.0	30265.1	0.29	8188	23	4487	18000
$\overrightarrow{Sun.dense}2$	29583.9	29715.8	29724.1	30647.1	30219.9	0.22	13626	46	1773	18000
$\overrightarrow{Sun.dense}3$	98575.5	98697.0	98707.0	99553	99329.7	0.06	13588	25	1850	18000
$\overrightarrow{Sun.dense}4$	98347.9	98466.5	98480	99343.8	99092.4	0.56	7789	-	555	18000
$\overrightarrow{Sun.dense}5$	59115.5	59228.8	59244.2	60157.9	59847.5	0.19	14649	26	1687	18000
$\overrightarrow{Sun.dense}6$	58923.4	59038.4	59051.2	60014.1	59667.5	0.08	11871	25	2627	18000

Table 5. Computational results for the $\overrightarrow{Sun.dense}$ instances.

41 nodes and 646 demand pairs and *Bar64*, a German instance with 64 nodes and 2016 demand pairs. From each of these two instances a subset of nodes was selected to form two backbone networks with, respectively, 13 and 9 nodes. We generated more backbone networks, defined by the subgraph induced by the nodes with largest demands, that will be made public available through a website.

Nevertheless we remark that *Bar41* and *Bar64* are still out of the scope of our algorithm as we were not able to solve the LP-relaxation. In table 1 we report on computational results for the backbone instances. We set a time limit of 10800 seconds for the search tree. Here the LP-based heuristic turned out to be ineffective and we used the re-routing heuristic, whose results are reported in the column UB_{Heu} .

Name	LB_{Flow}	LB_{Cuts}	BLB	UB_{Heu}	BUB	Gap%	#RM	#Mod- k	Nodes	Time
<i>Bar41/9</i>	155205	174517	176073	176385	176073	0.00	90	9	96	37
<i>Bar41/12</i>	251702	286060	291188	294196	291188	0.00	524	35	96	480
<i>Bar41/19</i>	307044	352559	359237	363945	363945	2.2	3268	74	685	12859
<i>Bar41/21</i>	316817	365739	368338	474059	474059	22.3	4297	121	2236	219
<i>Bar41/28</i>	383692	438536	438536	594692	594692	26.2	21238	194	2236	118500
<i>Bar64/13</i>	686817	889686	961550	977750	961550	0.00	2184	27	4711	3110
<i>Bar64/14</i>	705518	937792	977700	1011980	977700	0.00	696	38	1443	1008
<i>Bar64/17</i>	764169	995694	1061560	1099800	1099800	3.5	2332	52	5914	372
<i>Bar64/19</i>	807992	1070610	1134030	1623100	1623100	30.1	2886	71	2882	11172
<i>Bar64/26</i>	951923	1262090	1301920	2089430	2089430	37.7	2975	99	718	13137
<i>Bar64/32</i>	1076730	1477130	1495760	2406900	2406900	37.9	5693	125	137	12871

Table 6. Computational results for the *Bar64/XX* and *Bar41/XX* instances

References

1. A. Atamturk: On Capacitated Network Design Cut-Set Polyhedra, *Math. Progr.* **92** (2002) 425-437.
2. A. Atamturk, D. Rajan: On Splittable and Unsplittable Capacitated Network Design Arc-Set Polyhedra, *Math. Progr.* **92** (2002) 315-333.

3. A. Balakrishnan, T.L. Magnanti, P. Mirchandani: Network Design, Chapter 18 in *Annotated Bibliographies in Combinatorial Optimization*, M. Dell'Amico, F. Maffioli, and S. Martello (Eds.), Wiley (1997).
4. F. Barahona: Network Design Using Cut Inequalities, SIAM J. Optimization **6-3** (1996) 823-837.
5. D. Berger, B. Gendron, J.Y. Potvin, S. Raghavan, P. Soriano: Tabu Search for a Network Loading Problem with Multiple Facilities, J. of Heuristics (to appear) (1999).
6. D. Bienstock: Experiments with a Network Design Algorithm using ϵ -approximate Linear Programs, *manuscript* (1996).
7. D. Bienstock: Approximately solving large-scale linear programs. I: Strengthening lower bounds and accelerating convergence, *CORC Report* 1999-1, Columbia University (1999).
8. D. Bienstock: *Potential Function Methods for Approximately Solving Linear Programming Problems. Theory and Practice*, Kluwer, Boston. (2002)
9. D. Bienstock, A. Bley: Capacitated Network Design with Multicast Commodities, Proc. of 8th International Conference on Telecommunication Systems, Nashville, March 9-12, 2000
10. D. Bienstock, S. Chopra, O. Günlük, C. Tsai: Minimum Cost Capacity Installation for Multicommodity Network Flows, Math. Progr. **81** (1998) 177-199.
11. D. Bienstock, O. Günlük: Computational Experience with a Difficult Mixed-Integer Multicommodity Flow Problem, Math. Progr. **68** (1995) 213-237.
12. D. Bienstock, O. Günlük: Capacitated Network Design—Polyhedral Structure, and Computation, INFORMS Journal On Computing **8(3)** (1996) 243-259.
13. D. Bienstock, O. Raskina: Asymptotic Analysis of the Flow Deviation Method for the Maximum Concurrent Flow Problem, Math. Progr. **91** (2002) 379-392.
14. A. Caprara, M. Fischetti: $\{0, \frac{1}{2}\}$ -Chvátal-Gomory Cuts, Math. Progr. **74** (1996) 221-235.
15. A. Caprara, M. Fischetti, A.N. Letchford: On the Separation of Maximally Violated mod- k Cuts, Math. Progr. **87-1** (2000) 37-56.
16. S. Chopra, I. Gilboa, S.T. Sastry: Source Sink Flows with Capacity Installation in Batches, Disc. Appl. Math. **85** (1998), 165-192.
17. T.G. Crainic, A. Frangioni, B. Gendron: Multicommodity Capacitated Network Design in *Telecommunications Network Planning*, B. Sansó, P. Soriano (Eds.), Kluwer Academic (1998).
18. T.G. Crainic, A. Frangioni, B. Gendron: Bundle-Based Relaxation Methods for Multicommodity Capacitated Fixed Charge Network Design Problems, Discrete Applied Mathematics, to appear (1998).
19. T.G. Crainic, M. Gendreau: Cooperative Parallel Tabu Search for Capacitated Network Design, Centre de recherche sur les transports, Report CRT-98-71 (1998), Université de Montréal .
20. T.G. Crainic, M. Gendreau, J. Farvolden: Simplex-Based Tabu Search for the Multicommodity Capacitated Fixed Charge Network Design Problem, INFORMS Journal on Computing **12(3)** (2000) 223-236.
21. G. Dahl and M. Stoer: A cutting plane algorithm for multicommodity survivable network design problems, INFORMS Journal on Computing **10(1)** (1998) 1-11.
22. M. Deza. and M. Laurent: *Geometry of Cuts and Metrics*. Springer-Verlag, Berlin, 1997.
23. F. Eisenbrand: On the Membership Problem for the Elementary Closure of a Polyhedron, Combinatorica **19** (1999), 297-300.

24. L. Fleischer: Approximating Fractional Multicommodity Flows Independent of the Number of Commodities, SIAM Journal Discrete Mathematics **13(4)** (2000) 505-520.
25. O. Günlük: A Branch-and-Cut Algorithm for Capacitated Network Design Problems, Math. Progr. **86** (1999), 17-39.
26. O. Günlük: A new min-cut max-flow ratio for multicommodity flow problems, Proceedings of the IPCO 2002 Conference.
27. K. Holmberg, D. Yuan: A Lagrangean Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem, Operations Research **48-3** (2000) 461-481.
28. M. Iri: On an extension of the max-flow min-cut theorem to multicommodity flows, Journal of the Operations Research Society of Japan **13** (1971) 129-135.
29. A.N. Letchford: Totally Tight Chvátal-Gomory Cuts, Operations Research Letters **30(2)** (2002), 71-73.
30. M. Lomonosov: Feasibility conditions for multiflow problems, Discrete Mathematics (1982)
31. M. Lomonosov, A. Sebo: On the geodesic structure of graphs: a polyhedral approach to metric decomposition, Proceedings of IPCO (1993) 221-234.
32. T.L. Magnanti, P. Mirchandani, R. Vachani: Modeling and Solving the Core Capacitated Network Loading Problem, Operations Research **43(1)** (1995) 142-157.
33. T.L. Magnanti, P. Mirchandani, R. Vachani: Modeling and Solving the Two-Facility Capacitated Network Loading Problem, Operations Research **43(1)** (1995) 142-157.
34. K. Onaga, O. Kakusho: On feasibility conditions of multicommodity flows in networks. Transactions on Circuit Theory **CT-18(4)** (1971) 425-429.
35. M. Stoer, G. Dahl: A polyhedral approach to multicommodity network design, *Numerische Mathematik* **68** (1994) 149-167.
36. L.A. Wolsey: *Integer Programming*, Wiley (2002)

Valid Inequalities Based on Simple Mixed-Integer Sets

Sanjeeb Dash and Oktay Günlük

Mathematical Sciences Dept., IBM Research, Yorktown Heights, NY 10598
sanjeebd@us.ibm.com, oktay@watson.ibm.com

Abstract. In this paper we use facets of the convex hull of mixed-integer sets with two and three variables to derive valid inequalities for integer sets defined by a single equation. These inequalities also define facets of the master cyclic group polyhedron of Gomory. In particular, our inequalities generalize the 2slope facets of Araoz, Gomory, Johnson and Evans (2003). In addition, they dominate the strong fractional cuts of Letchford and Lodi (2002).

1 Introduction

An important technique in generating cutting planes for integer programs is to work with a single implied constraint, and to derive valid inequalities for the non-negative integral vectors satisfying the constraint. Suppose we express the constraint in the form $\sum_{j \in J} a_j x_j + y = b$, and let

$$Y^+ = \left\{ x \in Z^{|J|}, y \in Z : \sum_{j \in J} a_j x_j + y = b, \quad x \geq 0, y \geq 0 \right\}.$$

Valid inequalities for Y^+ can be used as cutting planes for the original integer program. One way to generate valid inequalities for Y^+ is to use facets of the *master cyclic group polyhedron*:

$$P(n, r) = \text{conv} \left\{ w \in Z^{n-1} : \sum_{i=1}^{n-1} i w_i \equiv r \pmod{n}, \quad w \geq 0 \right\} \quad (1)$$

where $n, r \in Z$ and $n > r > 0$ and $a \equiv b \pmod{n}$ means $a - b$ is divisible by n . For a set $S \subseteq R^n$, $\text{conv}(S)$ denotes the convex hull of vectors in S . $P(n, r)$ is closely related to a relaxation of Y^+ obtained by relaxing the non-negativity requirement on the y variable:

$$Y = \left\{ x \in Z^{|J|}, y \in Z : \sum_{j \in J} a_j x_j + y = b, \quad x \geq 0 \right\}. \quad (2)$$

Facets of $P(n, r)$ yield inequalities valid for Y , which are also valid for Y^+ . The polyhedral structure of $P(n, r)$ was initially studied in Gomory [11] and in

Gomory and Johnson [12,13]. More recent references include [15], [3], and [14]. Also see [1] for an introduction.

In this paper we study facets of $P(n, r)$ and valid inequalities for Y that can be derived from the following simple polyhedral sets:

$$Q^1 = \{x \in R, y \in Z : x + y \geq b, x \geq 0\}$$

and

$$Q^2 = \{x \in R, y, z \in Z : x + \alpha y + z \geq \beta, x, y \geq 0\},$$

and their relaxations obtained by allowing the integral variables in these models to be $(1/t)$ -integral for $t \in Z$. Our work is partly motivated by the work of Marchand and Wolsey[18], who derive the well-known Gomory mixed-integer cut (GMIC) using Q^1 . GMIC can also be derived from a facet of the master cyclic group polyhedron [11], and our interest in $P(n, r)$ is partly due to the practical importance of the GMIC, which is now routinely used in MIP software.

As discussed in Gomory and Johnson [12,13], there is a one-to-one correspondence between the facets of $P(n, r)$ and the facets of the mixed-integer set obtained by including continuous variables in $P(n, r)$. From this, one can show that facets of $P(n, r)$ yield cutting planes for general mixed-integer programs.

Most of the proofs have been omitted in this paper due to space restrictions, but they can be found in [8].

2 Scaled MIR Inequalities

The simple mixed-integer rounding (MIR) procedure provides a unifying framework to derive valid inequalities for mixed-integer sets. See [18] for examples. Given a valid inequality $x + y \geq b$ for a mixed-integer set $X \subset \{(x, y) : x \in R_+, y \in Z\}$, the MIR inequality $x \geq r(\lceil b \rceil - y)$ where $r = b - \lfloor b \rfloor$ is also valid for X , see [19].

The MIR inequality is facet defining for the set $\text{conv}(Q^1)$, and

$$\text{conv}(Q^1) = \{x, y \in R : x + y \geq b, x \geq r(\lceil b \rceil - y), x \geq 0\}.$$

We emphasize that the variable y in Q^1 can be negative. For any $t \in Z$ we next define a relaxation of Y by letting the y variable take on $1/t$ -integral values:

$$Y^t = \{x \in Z^{|J|}, ty \in Z : \sum_{j \in J} a_j x_j + y = b, x \geq 0\}.$$

Multiplying the equation by t and substituting $z^t = ty$, we observe that the resulting set has the same form as Y . Define \hat{c}^t to be $tc - \lfloor tc \rfloor$ for $c \in R$ and $t \in Z$. If tb is not integral, we can use the MIR inequality, as in the derivation of the GMIC in Marchand and Wolsey[18], to show that

$$\sum_{j \in J : \hat{a}_j^t < \hat{b}^t} \frac{\hat{a}_j^t}{\hat{b}^t} x_j + \sum_{j \in J : \hat{a}_j^t \geq \hat{b}^t} \frac{1 - \hat{a}_j^t}{1 - \hat{b}^t} x_j \geq 1 \quad (3)$$

is valid for Y^t . Therefore, inequality (3), which we call *the t-scaled MIR inequality* or *the t-MIR cut*, is valid for Y .

When applied to a row of the simplex tableau, the 1-scaled MIR inequality gives the well-known *Gomory mixed-integer cut*. When $t > 1$, the resulting inequality is referred to as the *k-cut* in Cornuéjols, Li and Vandenbussche [7]. We next define the *t*-scaled MIR function and show that there are only a small number of *t*-scaled MIR inequalities for Y .

Definition 1. For a non-zero integer t , the *t*-scaled MIR function with parameter b is defined as:

$$f^{t,b}(v) = \begin{cases} \hat{v}^t / \hat{b}^t & \text{if } \hat{v}^t < \hat{b}^t, \\ (1 - \hat{v}^t) / (1 - \hat{b}^t) & \text{if } \hat{v}^t \geq \hat{b}^t, \end{cases}$$

where $\hat{v}^t = tv - \lfloor tv \rfloor$ and $\hat{b}^t = tb - \lfloor tb \rfloor$.

Using this definition, the *t*-scaled MIR inequality (3) becomes

$$\sum_{j \in J} f^{t,b}(a_j) x_j \geq 1.$$

Lemma 1. Assume the inequality defining Y is rational, and let n be the smallest integer such that $nb \in Z$ and $na_j \in Z$ for all $j \in J$. Then, there are at most $\lfloor n/2 \rfloor$ distinct *t*-MIR cuts for Y . In particular, if $n > t > \lfloor n/2 \rfloor$ then the *t*-MIR cut is the same as the $(n-t)$ -MIR cut. \square

We note that Example 1 in Cornuéjols, Li and Vandenbussche [7] deals with the set $Y = \{x \in Z^2, y \in Z : 1.4x_1 + 0.1x_2 + y = 4.3, x \geq 0\}$ which has at most 5 distinct *t*-MIR cuts due to Lemma 1 because $n = 10$. This explains why the scaled cuts for t and $10 - t$ turn out to be identical in [7].

It is not difficult to see that the equation defining $P(n, r)$ can be written in the same form as Y and therefore *t*-MIR cuts can be applied to $P(n, r)$. Gomory [11] showed that the 1-MIR cut defines a facet of $P(n, r)$. Combining results on automorphisms and homomorphisms in [11], we can prove the following result.

Theorem 1. For every integer $t = 1, 2, \dots, \lfloor n/2 \rfloor$, such that tr is not an integral multiple of n , the *t*-MIR cut defines a facet of $P(n, r)$. \square

Gomory, Johnson and Evans [15] present results based on a *shooting experiment* of Gomory and identify the “important” facets of the cyclic group polyhedra for small n (i.e., $n \leq 20$). In particular, they present coefficients of 13 important facet defining inequalities for $P(10, 9)$, $P(20, 19)$, $P(15, 14)$, and $P(13, 12)$. Intriguingly, all of these facets are scaled MIR facets. In Table 2, we list these inequalities with the corresponding master polyhedra, and their scaling parameter.

Polyhedron	Reference in [15]	Relative importance of facet			
		1	2	3	4
P(10,9)	Table 2	5-MIR	2-MIR	4-MIR	-
P(20,19)	Table 3	10-MIR	5-MIR	4-MIR	-
P(15,14)	Figures 5-8	5-MIR	3-MIR	6-MIR	1-MIR
P(13,12)	Figures 9-11	1-MIR	2-MIR	6-MIR	-

Table 1. Important group facets are scaled MIR

3 A Simple Polyhedral Set

In this section we study simple three variable mixed-integer sets. We first look at the set

$$Q^{2+} = \{x \in R, y, z \in Z : x + \alpha y + z \geq \beta, x, y, z \geq 0\},$$

when $\alpha, \beta \in R$ and satisfy $1 > \beta > \alpha > 0$, and $\lceil \beta/\alpha \rceil > \beta/\alpha$. Note that variable z is required to be non-negative. In a recent study Agra and Constantino [2] study Q^{2+} when β is an arbitrary positive number and give a polynomial-time algorithm that enumerates all of its facets. Under our restrictions on α and β , it is possible to describe the convex hull of Q^{2+} explicitly.

Lemma 2. *The following inequalities are facet defining for $\text{conv}(Q^{2+})$, and together with the non-negativity requirements, give a complete description of $\text{conv}(Q^{2+})$:*

$$x + \alpha y + \beta z \geq \beta, \quad (4)$$

$$(1/(\beta - \alpha \lfloor \beta/\alpha \rfloor))x + y + \lceil \beta/\alpha \rceil z \geq \lceil \beta/\alpha \rceil. \quad (5)$$

Proof. (validity) Note that inequality (4) can be obtained by treating $x + \alpha y$ as a continuous variable and writing the MIR inequality based on $(x + \alpha y) + z \geq \beta$. To obtain inequality (5), we start with inequality (4), divide it by α and relax the resulting inequality as follows:

$$x/\alpha + y + \lceil \beta/\alpha \rceil z \geq \beta/\alpha \quad (6)$$

Writing the MIR inequality where x/α is treated as a continuous variable and $y + \lceil \beta/\alpha \rceil z$ is treated as an integer variable gives inequality (5). \square

Let Q^2 be the relaxation of Q^{2+} obtained by allowing the z variable to assume negative values:

$$Q^2 = \{x \in R, y, z \in Z : x + \alpha y + z \geq \beta, x, y \geq 0\}$$

and remember that $\beta, \alpha \in R$ satisfy $1 > \beta > \alpha > 0$, and $\lceil \beta/\alpha \rceil > \beta/\alpha$. Even though β is required to be less than 1 in Q^2 , the fact that z can take on negative values makes the set fairly general. We next show that (4) and (5) are facet defining for Q^2 under mild conditions. They are not necessarily sufficient to describe its convex hull.

Lemma 3. *Inequality (4) is facet defining for $\text{conv}(Q^2)$. In addition, inequality (5) is facet defining for Q^2 if $1/\alpha \geq \lceil \beta/\alpha \rceil$.*

Proof. (validity) Inequality (4) is valid as it is derived in the proof of Lemma 2 without assuming that z is non-negative. To see that inequality (5) is valid, notice that the following inequalities are valid for Q^2 :

$$(1/\alpha)x + y + (\beta/\alpha)z \geq \beta/\alpha,$$

$$(1/\alpha)x + y + (1/\alpha)z \geq \beta/\alpha,$$

and therefore, for any $\gamma \in R$ satisfying $1/\alpha \geq \gamma \geq \beta/\alpha$, the following inequality

$$(1/\alpha)x + y + \gamma z \geq \beta/\alpha$$

is also valid as it can be obtained as a convex combination of valid inequalities. As $1/\alpha \geq \lceil \beta/\alpha \rceil$ by assumption, inequality (6) is valid for Q^2 , and therefore so is inequality (5). \square

When $1/\alpha = \lceil \beta/\alpha \rceil$, inequality (6) is the same as $(1/\alpha)x + y + (1/\alpha)z \geq \beta/\alpha$, and hence inequality (5) is an MIR inequality obtained after scaling $x + ay + z \geq \beta$ by $1/\alpha$.

Inequality (5) leads to strong inequalities even when all variables are non-negative. Let

$$Q^3 = \{x \in R, y, z \in Z : x + ay + z \geq b, x, y, z \geq 0\},$$

where $b = \lfloor b \rfloor + \hat{b} > 1$ with $b \notin Z$ and $\hat{b} > a > 0$. Furthermore, assume that $1/a \geq \lceil \hat{b}/a \rceil > \hat{b}/a$. If we treat $x + ay$ as a continuous variable and z as an integral variable and apply the MIR procedure, we obtain $x + ay + \hat{b}z \geq \hat{b}\lceil b \rceil$ which can be relaxed to

$$(1/a)x + y + \lceil \hat{b}/a \rceil z \geq \hat{b}\lceil b \rceil / a. \quad (7)$$

The MIR procedure can be applied again by treating $(1/a)x$ as a continuous variable and $(y + \lceil \hat{b}/a \rceil z)$ as an integral variable. Another possibility is to view the initial inequality as $x + ay + (z - \lfloor b \rfloor) \geq \hat{b}$ and obtain $x + ay + \hat{b}(z - \lfloor b \rfloor) \geq \hat{b}$ as the first MIR inequality which then leads to

$$(1/a)x + y + \lceil \hat{b}/a \rceil (z - \lfloor b \rfloor) \geq \hat{b}/a, \quad (8)$$

via convex combinations, for the second MIR step. Inequality (8) is strictly stronger than inequality (7) as $\hat{b}\lceil b \rceil / a = \hat{b}/a + \hat{b}\lfloor b \rfloor / a < \hat{b}/a + \lceil \hat{b}/a \rceil \lfloor b \rfloor$. Therefore the MIR procedure based on inequality (8) gives a stronger valid inequality. We illustrate this in Example 1 and in Figure 1.

Example 1. Consider the set Q^3 with $a = 0.4$ and $b = 1.7$ and note that $1/a = 2.5 > \lceil \hat{b}/a \rceil = \lceil 0.7/0.4 \rceil = 2$. In this case, inequality (7) becomes $2.5x + y + 2z \geq 3.5$ whereas inequality (8) is $2.5x + y + 2z \geq 3.75$. The second step MIR inequality based on inequality (7) is $2.5x + 0.5y + z \geq 2$ which is weaker than the second step MIR inequality based on inequality (8): $2.5x + 0.75y + 1.5z \geq 3$. We also note this last inequality is facet defining for this instance. \square

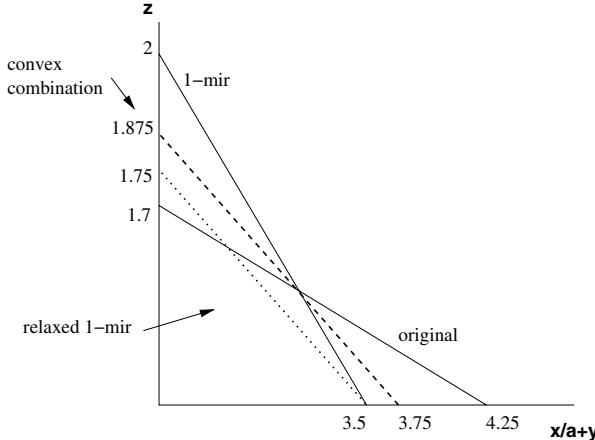


Fig. 1. Different relaxations of MIRs

In Figure 1, we plot four inequalities for the example above, by showing the boundaries of their feasible regions: the original inequality; the MIR inequality based on z being integral; inequality (7), shown by a dotted line; and inequality (8), shown by a dashed line. It is only when we apply the second MIR step that we distinguish between y and x ; therefore we can depict the above inequalities in the plane, the horizontal axis representing $2.5x + y$ and the vertical axis standing for z values.

4 Two-Step MIR Inequalities for Y and $P(n, r)$

In this section we apply inequality (5) to obtain valid inequalities for Y and facets of $P(n, r)$. Let $w = \sum_{j \in J} \lfloor a_j \rfloor x_j + y - \lfloor b \rfloor$ and $\hat{a}_j = a_j - \lfloor a_j \rfloor$, $\hat{b} = b - \lfloor b \rfloor$. The equality defining Y can now be written as

$$\sum_{j \in J} \hat{a}_j x_j + w = \hat{b}. \quad (9)$$

Note that w is integral provided that x and y are integral.

Let $\alpha \in R$ be such that $\hat{b} > \alpha > 0$, and $1/\alpha \geq \lceil \hat{b}/\alpha \rceil > \hat{b}/\alpha$. Let $\{J_1, J_2, J_3, J_4\}$ be a partition of the index set J such that (i) if $j \in J_2$, then $\hat{a}_j < \alpha$, and (ii) if $j \in J_3$, then $\hat{a}_j \geq \alpha$. We can now relax (9) to obtain

$$\sum_{j \in J_1} \hat{a}_j x_j + \sum_{j \in J_2} \alpha x_j + \sum_{j \in J_3} (\alpha + (\hat{a}_j - \alpha)) x_j + \sum_{j \in J_4} x_j + w \geq \hat{b}.$$

Rearranging the terms leads to

$$\left(\sum_{j \in J_1} \hat{a}_j x_j + \sum_{j \in J_3} (\hat{a}_j - \alpha) x_j \right) + \alpha \left(\sum_{j \in J_2 \cup J_3} x_j \right) + \left(\sum_{j \in J_4} x_j + w \right) \geq \hat{b},$$

which resembles the set Q^2 as the first term is non-negative, the second term gives positive integral multiples of α , and the last term is integral. As we chose α to satisfy $1/\alpha \geq \lceil \hat{b}/\alpha \rceil$, we can write the following valid inequality for Y based on inequality (5):

$$\sum_{j \in J_1} \hat{a}_j x_j + \sum_{j \in J_3} (\hat{a}_j - \alpha) x_j \geq \left(\hat{b} - \alpha \left\lceil \hat{b}/\alpha \right\rceil \right) \left(\left\lceil \hat{b}/\alpha \right\rceil - \sum_{j \in J_2 \cup J_3} x_j - \left\lceil \hat{b}/\alpha \right\rceil \left(\sum_{j \in J_4} x_j + w \right) \right).$$

Letting $\rho = \hat{b} - \alpha \lfloor \hat{b}/\alpha \rfloor$ and $\tau = \lceil \hat{b}/\alpha \rceil$, we can show that the strongest inequality of this form is obtained by partitioning J into $J_1^* = \{j \in J : \hat{a}_j < \rho\}$, $J_2^* = \{j \in J : \rho \leq \hat{a}_j < \alpha\}$, $J_3^* = \{j \in J : \alpha \leq \hat{a}_j < \rho(\tau - 1) + \alpha\}$, $J_4^* = \{j \in J : \rho(\tau - 1) + \alpha \geq \hat{a}_j\}$.

If $\tau > 2$, we strengthen the previous inequality by using a stronger relaxation of inequality (9) as the starting inequality. We partition the index set J into 2τ subsets $\{J_1^0, J_1^1, J_2^0, \dots, J_\tau^1\}$ as follows: For a given index $j \in J$, let $k_j = \min\{\lceil \hat{a}_j/\alpha \rceil, \tau\} - 1$. Index j is put in the set $J_{k_j}^0$ if $\hat{a}_j - k_j\alpha < \rho$; it is put in the set $J_{k_j}^1$ if $\hat{a}_j - k_j\alpha \geq \rho$. We now relax (9) to obtain

$$\sum_{j \in J_1^0} \hat{a}_j x_j + \sum_{k=1}^{\tau-1} \left(\sum_{j \in J_k^1} k\alpha x_j + \sum_{j \in J_{k+1}^0} (k\alpha + (\hat{a}_j - k\alpha)) x_j \right) + \sum_{j \in J_\tau^1} x_j + w \geq \hat{b},$$

which can be rewritten as

$$\left(\sum_{j \in J_1^0} \hat{a}_j x_j + \sum_{k=1}^{\tau-1} \sum_{j \in J_{k+1}^0} (\hat{a}_j - k\alpha) x_j \right) + \alpha \left(\sum_{k=1}^{\tau-1} \sum_{j \in J_k^1 \cup J_{k+1}^0} k x_j \right) + \left(\sum_{j \in J_\tau^1} x_j + w \right) \geq \hat{b}.$$

Applying inequality (5) and substituting for w leads to inequality $\sum_{j \in J} \gamma_j x_j - \rho \tau y \geq \rho \tau \lceil b \rceil$, where

$$\gamma_j = \rho \tau \lfloor a_j \rfloor + \begin{cases} (\hat{a}_j - k\alpha) + k\rho & \text{if } j \in J_{k+1}^0, k = 0, \dots, \tau - 1 \\ k\rho & \text{if } j \in J_k^1, k = 1, \dots, \tau. \end{cases}$$

We next define the two-step MIR function and formally state that the two-step MIR inequality is valid for Y .

Definition 2. Let $b, \alpha \in R$ be such that $\hat{b} > \alpha > 0$, and $1/\alpha \geq \lceil \hat{b}/\alpha \rceil > \hat{b}/\alpha$. Define $\rho = \hat{b} - \alpha \lfloor \hat{b}/\alpha \rfloor$, and $\tau = \lceil \hat{b}/\alpha \rceil$. The two-step MIR function for a right-hand side b with parameter α is defined by

$$g^{b,\alpha}(v) = \begin{cases} \frac{\hat{v}(1 - \rho\tau) - k(v)(\alpha - \rho)}{\rho\tau(1 - \hat{b})} & \text{if } \hat{v} - k(v)\alpha < \rho \\ \frac{k(v) + 1 - \tau\hat{v}}{\tau(1 - \hat{b})} & \text{if } \hat{v} - k(v)\alpha \geq \rho \end{cases}$$

where $k(v) = \min\{\lceil \hat{v}/\alpha \rceil, \tau\} - 1$, and $\hat{v} = v - \lfloor v \rfloor$ and $\hat{b} = b - \lfloor b \rfloor$.

Lemma 4. For any $\alpha \in R$ that satisfies the conditions in Definition 2, two-step MIR inequality for right-hand side b with parameter α

$$\sum_{j \in J} g^{b,\alpha}(a_j) x_j \geq 1 \quad (10)$$

is valid for Y . \square

Observe that $g^{b,\alpha}(v)$ is a piecewise linear function with two distinct slopes, $(1 - \rho\tau)/(\rho\tau(1 - \hat{b}))$ and $-1/(1 - \hat{b})$. Also, $1 \geq g^{b,\alpha}(v) \geq 0$, for all $v \in R$. In [13,14], Gomory and Johnson describe a family of piecewise linear “two-slope” functions, containing $g^{b,\alpha}(v)$, which yields valid inequalities for Y . Lemma 4 shows that, of the two-slope functions in [13], the ones of the form $g^{b,\alpha}(v)$ can be derived from the the two-step MIR inequalities.

It is possible to write t -scaled two-step MIR inequalities for Y by scaling the initial equality by $t \in Z$. For the sake of completeness, we next present the t -scaled two-step MIR inequalities for Y . Note that $1 > \hat{b}^t \geq 0$ even when $t < 0$.

Lemma 5. Let $t \in Z$ and $\alpha \in R$ be such that $\hat{b}^t > \alpha > 0$, and $1/\alpha \geq \lceil \hat{b}^t/\alpha \rceil > \hat{b}^t/\alpha$. Then, the t -scaled two-step MIR inequality $\sum_{j \in J} g^{tb,\alpha}(ta_j)x_j \geq 1$ is valid for Y . \square

When applied to $P(n, r)$, the two-step MIR inequalities derived in the previous section yield a wide range of facets. In [3], the authors present a class of facets of $P(n, r)$ which they call “2slope” facets. We will see that these 2slope facets are $(n - 1)$ -scaled (or (-1) -scaled) two-step MIR inequalities for appropriate choices of α . Initially we consider 1-scaled inequalities and present a result that generalizes Theorem 3.5 of [3]. We refer to the facets in Theorem 2 as *general two-slope* facets, in keeping with the notation in [3].

Theorem 2. Let $\Delta \in Z^+$ be such that $r > \Delta > 0$. The two-step MIR inequality

$$\sum_{i=1}^{n-1} g^{r/n, \Delta/n}(i/n) w_i \geq 1$$

defines a facet of $P(n, r)$ provided that $n > \Delta \lceil r/\Delta \rceil > r$. We call the corresponding facet the 1-scaled two-step MIR facet of $P(n, r)$ with parameter Δ .

We next show that t -scaled two-step MIR inequalities define facets of $P(n, r)$ under some conditions. Let $\mu_n^t(i) = ti \bmod n$ for integers i .

Theorem 3. Let $t \in \mathbb{Z}$ be such that tr is not divisible by n . Let $\Delta \in \mathbb{Z}^+$ be such that $\Delta = tk \bmod n$ for some integer k . The t -scaled two-step MIR inequality

$$\sum_{i=1}^n g^{tr/n, \Delta/n}(ti/n) w_i \geq 1$$

defines a facet of $P(n, r)$ provided that $\mu_n^t(r) > \Delta > 0$, and $n > \Delta \lceil \mu_n^t(r)/\Delta \rceil > \mu_n^t(r)$.

Based on this result, we can show the following result.

Corollary 1. The 2slope facets of $P(n, r)$ in [3, Theorem 3.4] with parameter d are $(n - 1)$ -scaled two-step MIR facets of $P(n, r)$ with parameter $\Delta = n - d$ such that $\tau = 2$, where $\tau = \lceil (n - r)/\Delta \rceil$.

Figure 2(a) shows a two-step MIR facet for $P(10, 7)$ obtained by letting $\Delta = 4$. Figure 2(b) shows a 3-scaled two-step MIR facet for $P(10, 9)$ obtained by setting $\Delta = 4$. In Figure 2, the marked points denote the coefficients of the corresponding facet-defining inequalities and the solid lines represent the underlying two-step MIR functions.

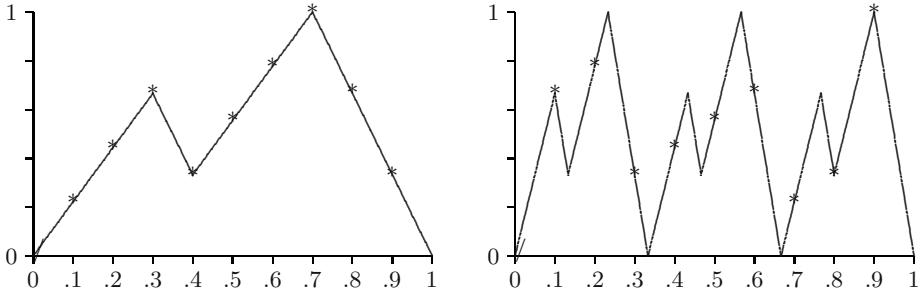


Fig. 2. Two different two-step MIR facets.

5 Extended Two-Step MIR Inequalities

In this section we derive new valid inequalities by considering the limiting behavior of the two-step MIR function $g^{b, \alpha}$. For a given right-hand side b , the set of admissible values of α that give valid inequalities for Y is not clear from Definition 2. Let $b \in R$ be fixed and let \mathcal{I} denote the set of values of α that satisfy the conditions of Definition 2. Let $\mathcal{I} = \cup_{d=2}^{\infty} \mathcal{I}^d$, where \mathcal{I}^d is the set of values of α such that $\lceil \hat{b}/\alpha \rceil = d$. In other words, $\mathcal{I}^d = (\hat{b}/d, 1/d] \cap (\hat{b}/d, \hat{b}/(d-1))$.

Notice that $\mathcal{I}^d \neq \emptyset$ for all $d \geq 2$. Also, $\alpha = 1/d$ is an admissible choice if and only if $1/d < \hat{b}/(d-1)$. Let \bar{d} be the largest integer strictly less than $1/(1-\hat{b})$,

that is, $\bar{d} = \lceil 1/(1 - \hat{b}) \rceil - 1$. Then, one can show that $\alpha = 1/d$ is an admissible choice in Lemma 4 only when $2 \leq d \leq \bar{d}$. Also,

$$\mathcal{I}^d = \begin{cases} (\hat{b}/d, 1/d] & \text{if } 2 \leq d \leq \bar{d} \\ (\hat{b}/d, \hat{b}/(d-1)) & \text{if } d > \bar{d}. \end{cases} \quad (11)$$

Let $\mathcal{E} = \{\hat{b}/d : d \geq \bar{d}, d \in \mathbb{Z}\}$ and notice that $\cup_{d=\bar{d}}^{\infty} \mathcal{I}^d = (0, 1/\bar{d}] \setminus \mathcal{E}$. Next, we will derive *extended two-step MIR inequalities* for Y , by considering the limiting behavior of the function $g^{b,\alpha}$ when α tends to a point p in the set \mathcal{E} . We will separately consider the case when α converges to p from below and from above, as the limits are different. We use $\epsilon \rightarrow 0^+$ to denote that ϵ takes only positive values as it tends to 0.

Definition 3. Let $b \in R$ be such that $\hat{b} > 0$. Let d be an integer such that $d \geq \lceil 1/(1 - \hat{b}) \rceil - 1$. The extended two-step MIR function for a right-hand side b with parameter d is defined by

$$h^{b,d}(v) = \begin{cases} \hat{v}/\hat{b} & \text{if } v \text{ is an integral multiple of } \hat{b}/d \text{ and } \hat{v} < \hat{b}, \\ \frac{l(v) + 1 - (d+1)\hat{v}}{(d+1)(1 - \hat{b})} & \text{otherwise,} \end{cases}$$

where $l(v) = \min\{\lfloor d\hat{v}/\hat{b} \rfloor, d\}$.

Lemma 6. For any integer d satisfying the conditions of Definition 3, and for any $v \in R$,

$$\lim_{\epsilon \rightarrow 0^+} g^{b,(\hat{b}/d)-\epsilon}(v) = h^{b,d}(v).$$

To complete the discussion on the limiting behavior of $g^{b,\alpha}$, we note that for any integer $d \geq \bar{d}$, the two-step MIR function converges to the (1-scaled) MIR function when α in $g^{b,\alpha}$ converges to a point in \mathcal{E} from above.

Let $\{a_i\} \subseteq R^n$ be a sequence converging to the vector a , and let $\{b_i\} \subseteq R$ be a sequence of numbers converging to b , such that $a_i^T x \leq b_i$ is valid for a polyhedron P , for all $i > 0$. Then $a^T x \leq b$ is also valid for P . This fact and Lemma 6 imply the following result.

Lemma 7. Let t be an arbitrary integer, and let d be a positive integer that satisfies the conditions in Definition 3 with b replaced by tb . Then the following inequality, called the t -scaled extended two-step MIR inequality, is valid for Y :

$$\sum_{j \in J} h^{tb,d}(ta_j) x_j \geq 1 \quad (12)$$

5.1 The Strong Fractional Cut of Letchford and Lodi

Let Y^+ be the restriction of the set Y obtained by requiring the y variable to be non-negative, i.e., $Y^+ = Y \cap \{y \geq 0\}$. In [16], Letchford and Lodi present a valid inequality for Y^+ which they call the *strong fractional cut*. Their inequality dominates the so-called *Gomory fractional cut*

$$\sum_{i \in J} \hat{a}_i x_i \geq \hat{b}. \quad (13)$$

In this section we will show that their inequality is implied by the extended two-step MIR inequalities. For convenience, we first present their main result in our notation.

Theorem 4. (Letchford and Lodi [16]) Suppose $\hat{b} > 0$ and let $k \geq 1$ be the unique integer such that $\frac{1}{k+1} \leq \hat{b} < \frac{1}{k}$. Partition J into classes Q_0, \dots, Q_k as follows. Let $Q_0 = \{i \in J : \hat{a}_i \leq \hat{b}\}$ and, for $p = 1, \dots, k$, let $Q_p = \{i \in J : \hat{b} + (p-1)(1-\hat{b})/k < \hat{a}_i \leq \hat{b} + p(1-\hat{b})/k\}$. Then the following strong fractional cut is valid for Y :

$$\sum_{i \in Q_0} \hat{a}_i x_i + \sum_{p=1}^k \sum_{i \in Q_p} (\hat{a}_i - p/(k+1)) x_i \geq \hat{b}. \quad (14)$$

It can be shown that $k = \lceil 1/\hat{b} \rceil - 1$, and that inequality (14) can be written as

$$\sum_{i \in J} \left(\hat{a}_i - \frac{1}{k+1} \max\{0, \left\lceil \frac{k(\hat{a}_i - \hat{b})}{1-\hat{b}} \right\rceil\} \right) x_i \geq \hat{b}. \quad (15)$$

We now show that the (-1)-scaled extended 2-step MIR inequality with parameter k dominates inequality (15). First consider a relaxation of $h^{b,d}$ defined by:

$$\tilde{h}^{b,d}(v) = \frac{l(v) + 1 - (d+1)\hat{v}}{(d+1)(1-\hat{b})}, \text{ where } l(v) = \min\{\left\lfloor d\hat{v}/\hat{b} \right\rfloor, d\}, \forall v \in R.$$

When \hat{v} is an integer multiple of \hat{b}/d and $\hat{v} < \hat{b}$, we know that $h^{b,d}(v) = \hat{v}/\hat{b}$, and therefore $h^{b,d}(v) < \tilde{h}^{b,d}(v)$. For all other values of v , $h^{b,d}(v)$ and $\tilde{h}^{b,d}(v)$ are identical. Now, multiply the equation defining the set Y^+ by -1 and apply $\tilde{h}^{-b,k}$, where $k = \lceil 1/\hat{b} \rceil - 1$, to get the valid inequality $\sum_{j \in J} \tilde{h}^{-b,k}(-a_j)x_j \geq 1$.

As $(-a_j) - \lfloor -a_j \rfloor = 1 - \hat{a}_j$ and $(-b) - \lfloor -b \rfloor = 1 - \hat{b}$, this is:

$$\sum_{i \in J} \frac{\min\{k, \left\lfloor \frac{k(1-\hat{a}_i)}{1-\hat{b}} \right\rfloor\} + 1 - (k+1)(1-\hat{a}_i)}{(k+1)\hat{b}} x_i \geq 1$$

By simple algebraic transformations, the above inequality is the same as inequality inequality (15).

Recall \bar{d} in Definition 3. As we are working with the equation defining Y^+ scaled by -1 , k in (15) is actually equal to \bar{d} . This implies the following result:

Proposition 1. *The strong fractional cut (14) of Theorem 4 is dominated by the (-1) -scaled extended two-step MIR inequality (12). Furthermore, inequality (14) is valid for all $k \geq \lceil 1/\hat{b} \rceil - 1$.*

6 Concluding Remarks

In this paper we derived the two-step MIR inequalities (10) and showed how these inequalities generalize both the 2slope facets of $P(n, r)$ in [3] and the strong fractional cuts in [16]. In deriving these inequalities, we considered only one of the many facets of the set Q^2 , and applied this facet to a specific relaxation of Y . It is somewhat surprising that such a seemingly narrow procedure results in such general inequalities.

It is relatively straightforward to generalize the two-step MIR inequalities when the equation defining the set Y contains continuous non-negative variables in addition to integral variables. An important question is whether these inequalities will be useful for solving general mixed-integer programs, the way GMICs are. Based on the observation that the two-step MIR inequalities have a number of features in common with the GMIC, we are optimistic. In particular, we note they are both based on facets of simple mixed-integer sets, and they define important facets of low-dimensional cyclic group polyhedra.

Acknowledgments: We would like to thank Jon Lee for his comments on an earlier version of paper. We would also like to thank Lisa Evans Miller for fruitful discussions on scaled MIR inequalities for $P(n, r)$.

References

1. K. Aardal, R. Weismantel, and L.A. Wolsey, Non-standard approaches to integer programming, *Discrete Applied Mathematics* **123** 5–74 (2002).
2. A. Agra and M. F. Constantino, Description of 2-integer continuous knapsack polyhedra, working paper (2003).
3. J. Araoz, R.E. Gomory, E.L. Johnson, and L. Evans, Cyclic group and knapsack facets, *Mathematical Programming* **96** 377–408 (2003).
4. E. Balas, S. Ceria, G. Cornuéjols, A lift-and-project cutting plane algorithm for mixed 0-1 programs, *Mathematical Programming* **58** 295–324 (1993).
5. E. Balas, S. Ceria, G. Cornuéjols, G. Natraj, Gomory cuts revisited, *Operations Research Letters* **19** 1–9 (1996).
6. R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and practice – closing the gap. In M. J. D. Powell and S. Scholtes, editors, *System Modelling and Optimization: Methods, Theory, and Applications*, pages 19–49. Kluwer Academic Publishers, 2000.

7. G. Cornuejols, Y. Li and D. Vandenbussche, K-Cuts: A variation of Gomory mixed integer cuts from the LP tableau, To appear in *INFORMS Journal on Computing*.
8. S. Dash and O. Günlük, Valid Inequalities Based on Simple Mixed-Integer Sets, IBM Research Report RC22922 (2003).
9. S. Dash and O. Günlük, Comparing valid inequalities for cyclic group polyhedra, IBM Research Report RC22989 (2003).
10. L. Evans, Cyclic groups and knapsack facets with applications to cutting planes, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, Georgia, 2002.
11. R.E. Gomory, Some polyhedra related to combinatorial problems, *Journal of Linear Algebra and its Applications*, **2**, 451–558 (1969).
12. R.E. Gomory and E. Johnson, Some continuous functions related to corner polyhedra I, *Mathematical Programming* **3** 23–85 (1972).
13. R.E. Gomory and E. Johnson, Some continuous functions related to corner polyhedra II, *Mathematical Programming* **3** 359–389 (1972).
14. R.E. Gomory and E. Johnson, T-space and cutting planes, *Mathematical Programming* **96** 341–375 (2003).
15. R.E. Gomory, E.L. Johnson, and L. Evans, Corner Polyhedra and their connection with cutting planes, *Mathematical Programming* **96** 321–339 (2003).
16. A. N. Letchford and A. Lodi, Strengthening Chvátal-Gomory cuts and Gomory fractional cuts, *Operations Research Letters* **30**(2) 74–82 (2002).
17. T.L. Magnanti, P. Mirchandani, and R. Vachani, The convex hull of two core capacitated network design problems, *Math. Programming* **60**, 233–250 (1993).
18. H. Marchand and L. Wolsey, Aggregation and mixed integer rounding to solve MIPs, *Oper. Res.* **49**, 363–371 (2001).
19. G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York (1988).

The Price of Anarchy when Costs Are Non-separable and Asymmetric

G. Perakis

Massachusetts Institute of Technology, Cambridge MA 02139,
georgiap@mit.edu

Abstract. In this paper we characterize the “price of anarchy”, i.e., the inefficiency between user and system optimal solutions, when costs are non-separable, asymmetric and nonlinear, generalizing earlier work that has addressed “the price of anarchy” under separable costs. This generalization models traffic equilibria, competitive multi-period pricing and competitive supply chains. The bounds established in this paper are tight and explicitly account for the degree of asymmetry and nonlinearity of the cost function. We introduce an alternate proof method for providing bounds that uses ideas from semidefinite optimization. Finally, in the context of multi-period pricing our analysis establishes that user and system optimal solutions coincide.

Keywords: System and User-Optimization, Traffic Equilibrium, Price of Anarchy.

1 Introduction

There has been an increasing literature in the recent years trying to quantify the inefficiency of Nash equilibrium problems (user-optimization) in non-cooperative games. The fact that there is not full efficiency in the system is well known both in the economics but also in the transportation literature (see [1], [8]). This inefficiency of user-optimization was first quantified by Papadimitriou and Koutsoupias [15] in the context of a load balancing game. They coined the term “the price of anarchy” for characterizing the degree of efficiency loss. Subsequently, Roughgarden and Tardos [21] and Roughgarden [20] applied this idea to the classical network equilibrium problem in transportation with arc cost functions that are separable in terms of the arc flows. They established worst case bounds for measuring this inefficiency for affine separable cost functions and subsequently for special classes of separable nonlinear ones (such as polynomials). It should be noted that Marcotte presented in [16], results on the “price of anarchy” for a bilevel network design model. Recently, Johari and Tsitsiklis [14] also studied this problem in the context of resource allocation between users sharing a common resource. In their case the problem also reduces to one where each player has a separable payoff function. Correa, Schulz and Stier Moses [4] have also studied “the price of anarchy” in the context of transportation for capacitated

networks. The cost functions they consider are also separable. The paper by Chau and Sim [2] has recently considered the case of nonseparable, symmetric cost functions giving rise to the same bound as Roughgarden and Tardos [21]. Wardrop [24] was perhaps the first to state the equilibrium principles in the context of transportation. Dafermos and Sparrow [5] coined the terms “user-optimized” and “system-optimized” in order to distinguish between Nash equilibrium where users act unilaterally in their own self interest versus when users are forced to select the routes that optimize the total network efficiency. Smith [22] and Dafermos [7] recognized that this problem can be cast as a variational inequality. In [6] Dafermos considered how the decentralized “user-optimized” problem can become a centralized “system optimization” problem through the imposition of tolls. Recently, Hearn and co-authors [12], [13], have studied the problem of imposing tolls in order to induce a behavior to users so that their route choices are optimizing the overall system. They study a variety of criteria for imposing tolls. Finally, Cole, Dodis and Roughgarden [3] have also studied the notion of introducing tolls (taxes) in order to make the decentralized problem efficient in a centralized manner. The review paper by Florian and Hearn [10], the book by Nagurney [17], and the references therein summarize the relevant literature in traffic equilibrium problems. Traffic equilibrium problems are typically modeled through variational inequalities. The books by Facchinei and Pang [9] summarize the developments in the area of variational inequalities.

Nash equilibrium problems arise in a variety of settings and model competitive and non-cooperative behavior (see Section 2 for more details). In this paper we study the inefficiency of equilibrium by comparing how the presence of competition affects the total profit in the system in a decentralized (user-optimized) versus a centralized optimization (system-optimized) setting. We establish a bound on the ratio of the overall profit of the system in these two settings. This work is the first to consider non-separable, asymmetric cost functions and is important since, as we discuss in Section 2, it allows modeling more realistic situations. Furthermore, it allows a unifying framework which naturally extends results in the current literature. In particular, our contributions versus the existing literature are the following.

1. We consider **non-separable** functions in the sense that cost functions also depend on the strategies of the competitors. Furthermore, cost functions can be **asymmetric** in the sense that different competitors’ strategies affect their cost functions differently. This generalization is important since the strategies of one’s competitors will influence his/her own cost in an asymmetric way. In particular, we introduce a measure of asymmetry (denoted by c^2 in Section 2) which quantifies **the degree of asymmetry** of the competitors’ cost functions. We establish that the ratio of the total cost in the system operating in a user-optimized setting versus the total cost in a system optimized setting is bounded by

$$\begin{cases} \frac{4}{4 - c^2} & \text{if } c^2 \leq 2 \\ c^2 & \text{if } c^2 > 2. \end{cases}$$

We illustrate how our results are a natural generalization of the bound established by [20], [21] since when the problem functions are separable and affine the bound becomes $4/3$. Furthermore, the work by Chau and Sim [2] (where the functions are non-separable but symmetric) becomes a special case since the bound is still $4/3$, since $c^2 = 1$. The results in the affine case allow the feasible region to be a non-convex set.

2. When the cost function has no constant term then the bound becomes c^2 . An important implication of this result arises in the context of multi-period pricing. In this setting our analysis establishes that user and system optimal solutions coincide.
3. We generalize our results to **nonlinear** functions. We introduce a measure which quantifies **the degree of nonlinearity** of the problem function (denoted by A in Section 3). We establish that the bound naturally extends to involve the nonlinearity parameter A , i.e.,

$$\begin{cases} \frac{4}{4 - c^2 A} & \text{if } c^2 \leq \frac{2}{A} \\ c^2 A^2 - 2(A - 1) & \text{if } c^2 > \frac{2}{A}. \end{cases}$$

4. We establish that the bounds above are **tight** for affine as well as for some nonlinear problems.
5. We introduce an alternative **semidefinite optimization** formulation for deriving these bounds. This approach does not require positive definiteness of the Jacobian matrix (i.e., it does not need to be invertible). Therefore, the solution does not need to be unique. We illustrate that this approach gives rise to the same bound when the Jacobian matrix is positive definite.

2 A Bound for Affine and Asymmetric Cost Functions

In this section, we establish a bound between the user and the system optimization problems (i.e. decentralized versus centralized problems) in the context of minimizing cost. For the (UO) decentralized problem we will consider the variational inequality problem of finding $x_u \in K$ satisfying

$$F(x_u)^t(x - x_u) \geq 0, \quad \text{for all } x \in K.$$

Let x_u and x_s denote solutions of the user and system optimization problems respectively. Let $Z_u = F(x_u)^t x_u$ be the total cost for the user-optimized problem (UO) and $Z_s = F(x_s)^t x_s = \min_{x \in K} F(x)^t x$ be the total cost for the system-optimized problem (SO). In this section, we provide a bound on Z_u/Z_s for cost functions $F(x) = Gx + b$, with $G \succ 0$ (i.e. positive definite) and asymmetric matrix, $b^t x \geq 0$ for all $x \in K$ (notice that this follows when constant vector $b \geq 0$ and $K \subseteq R_+^n$). In this case, the system optimization problem involves the minimization of a strictly convex quadratic function over the set K .

2.1 A Measure of Asymmetry of a Matrix

For a matrix G , we consider the symmetrized matrix

$$S = \frac{G + G^t}{2}$$

and introduce the following measure c^2 of the degree of asymmetry of matrix G :

Definition 1.

$$c^2 \equiv \|S^{-1}G\|_S^2 = \sup_{w \neq 0} \frac{\|S^{-1}Gw\|_S^2}{\|w\|_S^2} = \sup_{w \neq 0} \frac{w^t G^t S^{-1} G w}{w^t S w} = \|S^{-1/2} G S^{-1/2}\|^2.$$

Note that by setting $l = S^{1/2}w$, the previous definition of c^2 becomes

$$c^2 = \sup_{l \neq 0} \frac{l^t S^{-1/2} G^t S^{-1} G S^{-1/2} l}{\|l\|^2} = \lambda_{\max}(S^{-1/2} G^t S^{-1} G S^{-1/2}).$$

When the matrix G is positive definite and symmetric, that is, $G = G^t$ (and therefore, $S = G$), then $c^2 = 1$. As an example, consider

$$G = \begin{bmatrix} 1 & a \\ -a & 1 \end{bmatrix}.$$

Since $S = I$, it easily follows that $c^2 = 1 + a^2$. The quantity c^2 in this case quantifies the observation that as $|a|$ increases, the degree of asymmetry of G increases as well.

The next lemma indicates that if $G^2 \succeq 0$, then the degree of asymmetry of G is bounded.

Lemma 1. (see [11]) *If G^2 is a positive semidefinite matrix then $c^2 = \|S^{-1}G\|_S^2 \leq 2$.*

2.2 The Bound

Theorem 1. *For an affine variational inequality problem with problem function $F(x) = Gx + b$, with $G \succ 0$, $b^t x \geq 0$ for all $x \in K$, we have:*

$$\frac{Z_u}{Z_s} \leq \begin{cases} \frac{4}{4 - c^2} & \text{if } c^2 \leq 2 \\ c^2 & \text{if } c^2 > 2. \end{cases}$$

Proof:

We first observe that since x_u solves $VI(F, K)$ and $x_s \in K$ then the variational inequality formulation implies that

$$\begin{aligned}
Z_u &= F(x_u)^t x_u \leq F(x_u)^t x_s \\
&= x_u^t G^t x_s + b^t x_s \\
&= x_u^t G^t S^{-1} S x_s + b^t x_s \quad (\text{multiplying with } S \text{ and } S^{-1}) \\
&\leq \|x_u\|^t G^t S^{-1}\|_S \|x_s\|_S + b^t x_s \quad (\text{from Cauchy's inequality, and } S \succ 0) \\
&\leq \|x_u\|_S \|S^{-1}G\|_S \|x_s\|_S + b^t x_s \quad (\text{from the norm inequality}) \\
&= c \|x_u\|_S \|x_s\|_S + b^t x_s. \quad (\text{Definition 1})
\end{aligned}$$

For every $a_1, a_2 \geq 0$, we have

$$0 \leq (\sqrt{a_1} \|x_u\|_S - \sqrt{a_2} \|x_s\|_S)^2 = a_1 \|x_u\|_S^2 + a_2 \|x_s\|_S^2 - 2\sqrt{a_1 a_2} \|x_u\|_S \|x_s\|_S$$

leading to

$$2\sqrt{a_1 a_2} \|x_u\|_S \|x_s\|_S \leq a_1 \|x_u\|_S^2 + a_2 \|x_s\|_S^2.$$

This in turn implies that if we choose $2\sqrt{a_1 a_2} \geq c$, then

$$c \|x_u\|_S \|x_s\|_S \leq a_1 \|x_u\|_S^2 + a_2 \|x_s\|_S^2.$$

We thus obtain that for all $a_1, a_2 \geq 0$ and $a_1 a_2 \geq \frac{c^2}{4}$:

$$\begin{aligned}
Z_u &\leq a_1 \|x_u\|_S^2 + a_2 \|x_s\|_S^2 + b^t x_s \\
&= a_1 x_u^t S x_u + a_2 x_s^t S x_s + b^t x_s \\
&= a_1 (x_u^t G^t x_u + b^t x_u) + a_2 (x_s^t G^t x_s + b^t x_s) - a_1 b^t x_u - (a_2 - 1) b^t x_s.
\end{aligned}$$

If we further select $a_2 \geq 1$ and since $b^t x_u, b^t x_s \geq 0$, we obtain

$$Z_u \leq a_1 Z_u + a_2 Z_s.$$

If we further impose the condition $a_1 \leq 1$, we obtain the bound:

$$\frac{Z_u}{Z_s} \leq \frac{a_2}{1 - a_1}.$$

Given that we have freedom to select a_1 and a_2 , we find the best upper bound by solving

$$\begin{aligned}
&\text{minimize} \quad \frac{a_2}{1 - a_1} \\
&\text{subject to} \quad a_1 a_2 \geq \frac{c^2}{4} \\
&\quad a_2 \geq 1, \quad 0 \leq a_1 \leq 1.
\end{aligned} \tag{1}$$

The optimal solution to Problem (1) is given as follows:

If $c^2 \leq 2$, the optimal solution is $a_1 = c^2/4$, $a_2 = 1$ leading to

$$\frac{Z_u}{Z_s} \leq \frac{4}{4 - c^2}.$$

If $c^2 > 2$, the optimal solution is $a_1 = 1/2$, $a_2 = c^2/2$ leading to

$$\frac{Z_u}{Z_s} \leq c^2. \square$$

Remark: Notice that the results in this section apply to a feasible region that is non-convex and as a result includes integers.

(a) Separable affine functions:

When the variational inequality problem function F is separable, it has components $F_i(x) = g_i x_i + b_i$. In this case the matrix G is diagonal, with diagonal elements $g_i > 0$. In this case $c^2 = 1$ and the bound in Theorem 1 becomes

$$\frac{Z_u}{Z_s} \leq \frac{4}{4 - c^2} = \frac{4}{3}$$

originally obtained in Roughgarden and Tardos [21].

(b) Non-separable symmetric affine functions:

When the variational inequality problem function F is non-separable, that is $F(x) = Gx + b$, with G a general symmetric positive definite matrix, then $c^2 = 1$ and thus $Z_u/Z_s \leq 4/3$. This illustrates that the bound of $4/3$ by Chau and Sim [2] is also a special case of the bound in this paper.

(c) Non-separable asymmetric affine functions:

When the matrix G is “not too asymmetric” (in the sense that $c^2 \leq 2$) then the bound becomes $\frac{4}{4 - c^2}$. On the other hand, for “rather asymmetric” matrices (in the sense that $c^2 > 2$) then the bound becomes c^2 .

Corollary 1. (see [19]) *When the constant term is zero in the variational inequality problem function, that is, $F(x) = Gx$, where G is a positive definite asymmetric matrix, then*

$$\frac{Z_u}{Z_s} \leq c^2.$$

We now discuss some examples that illustrate the tightness of the bounds we established. In this subsection we discuss some examples that illustrate the tightness of the bounds we established. In particular through the example below we illustrate how the bound becomes tight for some families of problems.

Example:

Consider $F_1(x) = ax^1 + fx^2 + b_1$, $F_2(x) = -fx^1 + gx^2 + b_2$ and feasible region $K = \{x = (x^1, x^2) : x^1 + x^2 = d, x^1, x^2 \geq 0\}$.

Notice that the degree of asymmetry constant $c^2 = \lambda_{\max}(S^{-1/2}G^t S^{-1}GS^{-1/2}) = 1 + \frac{f^2}{ag}$, since in this example $G = \begin{pmatrix} a & f \\ -f & g \end{pmatrix}$, $S = \begin{pmatrix} a & 0 \\ 0 & g \end{pmatrix}$.

We will consider the following cases.

- (a)** First we consider the case where $a \leq g$. Let $f = a$ and constant terms $b_1 = b = (g - a)d$ and $b_2 = 0$. Notice that the measure of asymmetry becomes $c^2 = 1 + \frac{a}{g} \leq 2$, since $\frac{a}{g} \leq 1$. Furthermore, since $a \leq g$ then $b_1 = b \geq 0$. The user-optimized solution is $x_u^1 = 0$, $x_u^2 = d$ and $Z_u = gd^2$. The system-optimized solution is $x_s^1 = \frac{2gd-b}{2(a+g)} \geq 0$ (since $b = (g - a)d$), $x_s^2 = \frac{2ad+b}{2(a+g)} \geq 0$ and $Z_s = \frac{(3g-a)d^2}{4}$. Notice that then

$$\frac{Z_u}{Z_s} = \frac{4g}{3g - a} = \frac{4}{3 - \frac{a}{g}} = \frac{4}{4 - c^2}.$$

Furthermore, when in particular, $a = g$ then $c^2 = 2$ and $\frac{Z_u}{Z_s} = \frac{4}{4-c^2} = 2 = c^2$.

- (b) Consider the case where $a > g$. Choose as $f = -a$ and as constant terms $b_1 = 0$, $b_2 = b = (a-g)d$. Notice that since $a > g$, $b_2 = b > 0$. Moreover, the measure of asymmetry becomes $c^2 = 1 + \frac{a}{g} > 2$. The user-optimized solution is $x_u^1 = d$, $x_u^2 = 0$ and $Z_u = ad^2$. The system-optimized solution is $x_s^1 = \frac{2gd+b}{2(a+g)} \geq 0$, $x_s^2 = \frac{2ad-b}{2(a+g)} \geq 0$ (since $b_2 = b = (a-g)d$) and $Z_s = a(\frac{2gd+b}{2(a+g)})^2 + g(\frac{2ad-b}{2(a+g)})^2 + b\frac{2ad-b}{2(a+g)}$. Suppose that $d \rightarrow \infty$. Notice that then as $d \rightarrow \infty$,

$$\frac{Z_u}{Z_s} \longrightarrow \frac{a+g}{g} = 1 + \frac{a}{g} = c^2.$$

These two examples establish that the bound in Theorem 1 is tight. Furthermore, this last example also suggests that the bound is tight even when the demand d is very large.

- (c) Finally, consider the case where the constant terms $b_1 = b_2 = 0$. Choose as $f = -a$. Then $c^2 = 1 + \frac{a}{g}$. The user-optimized solution is $x_u^1 = d$, $x_u^2 = 0$ and $Z_u = ad^2$. The system-optimized solution is $x_s^1 = \frac{gd}{(a+g)} \geq 0$, $x_s^2 = \frac{ad}{(a+g)} \geq 0$ and $Z_s = \frac{agd^2}{a+g}$. Notice that then

$$\frac{Z_u}{Z_s} = \frac{a+g}{g} = 1 + \frac{a}{g} = c^2.$$

This example establishes that the bound in Corollary 1 is tight (i.e., when the constant term is zero).

In this example $G = \begin{pmatrix} a & f \\ -f & g \end{pmatrix}$ is a positive definite matrix since its symmetrized version is matrix $S = \frac{G+G^t}{2} = \begin{pmatrix} a & 0 \\ 0 & g \end{pmatrix}$. Furthermore, matrix G^2 becomes $\begin{pmatrix} a^2 - f^2 & af + fg \\ -af - fg & g^2 - f^2 \end{pmatrix}$ with a symmetrized matrix $\frac{G^2+(G^2)^t}{2} = \begin{pmatrix} a^2 - f^2 & 0 \\ 0 & g^2 - f^2 \end{pmatrix}$. Since in the family of examples above, we chose $f = a$ or $f = -a$, we observe that in both cases the symmetrized matrix of G^2 is $\begin{pmatrix} 0 & 0 \\ 0 & g^2 - a^2 \end{pmatrix}$. Therefore, G^2 is a positive semidefinite matrix when $a \leq g$. This is equivalent to $w^t G^2 w = (G^t w)^t (Gw) \geq 0$, which suggests that the angle between vectors $(G^t w)$ and (Gw) is less than or equal to 90 degrees. Notice that in this case it is also true that $c^2 = 1 + \frac{a}{g} \leq 2$. Furthermore, G^2 is a negative definite matrix when $a > g$. Since $w^t G^2 w = (G^t w)^t (Gw) < 0$, it follows that the angle between vectors $(G^t w)$ and (Gw) is more than 90 degrees. It is also the case that $c^2 = 1 + \frac{a}{g} > 2$. This latter observation illustrates further how the bound on constant c^2 connects with the degree of asymmetry of matrix G .

3 A Bound for Nonlinear, Asymmetric Functions

In this section, we assume that the Jacobian matrix is not a constant matrix G but a positive definite, nonlinear and asymmetric matrix $\nabla F(x)$. The positive definiteness assumption of the Jacobian matrix implies that the variational inequality problem has a unique solution (see for example [17] for details).

We introduce the symmetrized matrix, $S(x) = \frac{\nabla F(x) + \nabla F(x)^t}{2}$. We now extend Definition 1 for measuring the **degree of asymmetry** of the problem to a definition that also incorporates the nonlinearity of the cost functions involved.

Definition 2. *We define a quantity c^2 that measures the degree of asymmetry of the Jacobian matrix $\nabla F(x)$. That is,*

$$c^2 \equiv \sup_{x \in K} \|S(x)^{-1} \nabla F(x)\|_{S(x)}^2.$$

Constant c^2 is in this case the supremum over the feasible region, of the maximum eigenvalue of the positive definite and symmetric matrix

$$S(x)^{-1/2} \nabla F(x)^t S(x)^{-1} \nabla F(x) S(x)^{-1/2}.$$

When the Jacobian matrix is positive definite and symmetric, then $c^2 = 1$.

Furthermore, we need to define a measure of the **nonlinearity** of the problem function F . As a result, we consider a property of the Jacobian matrix which always applies to positive definite matrices. This allows us in some cases to provide a tight bound. This bound naturally extends the bound in Theorem 1 from affine to nonlinear problems. The bound involves the constant A that measure the nonlinearity of the problem.

Definition 3. (see [23] for more details)

The variational inequality problem function $F : R^n \rightarrow R^n$ satisfies **Jacobian similarity** property if it has a positive semidefinite Jacobian matrix ($\nabla F(x) \succeq 0, \forall x \in K$) and $\forall w \in R^n$, and $\forall x, \bar{x} \in K$, there exists $A \geq 1$ satisfying

$$\frac{1}{A} w^t \nabla F(x) w \leq w^t \nabla F(\bar{x}) w \leq A w^t \nabla F(x) w.$$

Lemma 2. (see [23]) The Jacobian similarity property holds under either of the following conditions:

- The Jacobian matrix is **strongly positive definite** (i.e. has eigenvalues bounded away from zero). Then a possible bound for the constant A is

$$A = \frac{\max_{x \in K} \lambda_{\max}(S(x))}{\min_{x \in K} \lambda_{\min}(S(x))}.$$

- The problem function is **affine**, with **positive semidefinite** Jacobian matrix G . In this case $A = 1$.

Theorem 2. (see [19]) For a variational inequality problem with a strictly monotone, nonlinear continuously differentiable problem function F satisfying the Jacobian similarity property, $F(0)^t x \geq 0$ for all $x \in K$, we have:

$$\frac{Z_u}{Z_s} \leq \begin{cases} \frac{4}{4 - c^2 A} & \text{if } c^2 \leq \frac{2}{A} \\ c^2 A^2 - 2(A - 1) & \text{if } c^2 > \frac{2}{A}. \end{cases}$$

Remarks:

1. When the variational inequality problem function is affine, $F(x) = Gx + b$, then $\nabla F(x) = G$ and as a result Definition 3 holds for $A = 1$. Therefore, $\frac{2}{A} = 2$ and the bound coincides with the one we found in the previous section.
2. When the variational inequality problem function has a symmetric Jacobian matrix, as we discussed $c^2 = 1$. Therefore,
 - i. If $A \leq 2$ it follows that $c^2 = 1 \leq \frac{2}{A}$ and $\frac{Z_u}{Z_s} \leq \frac{4}{4-A}$.
 - ii. If $A > 2$ it follows that $c^2 = 1 > \frac{2}{A}$ and $\frac{Z_u}{Z_s} \leq 1 + A^2$.
3. When the term $F(0) = 0$ then similarly to Corollary 1, the bound becomes $\frac{Z_u}{Z_s} \leq c^2 A^2 - 2(A - 1)$.

We discuss an example where the bound we established for nonlinear problems is tight.

Example:

Consider the feasible region

$$K = \{x = (x^1, x^2) \in R_+^2 : x^1 + x^2 = 1/2\}.$$

Consider cost functions

$$F_1(x) = (x^1)^2 + x^1 + ax^2, \quad F_2(x) = (x^2)^2 + x^2 - ax^1.$$

The Jacobian matrix is

$$\nabla F(x) = \begin{pmatrix} 2x^1 + 1 & a \\ -a & 2x^2 + 1 \end{pmatrix}.$$

In this example $A = \frac{\max_{x \in K} \lambda_{\max}(x)}{\min_{x \in K} \lambda_{\min}(x)} = \frac{2}{1} = 2$. The constant $c^2 = 1 + a^2$. The system-optimized solution is $x_s = (1/2, 1/2)$ which gives rise to $Z_s = F(x_s)^t x_s = \frac{3}{4}$. Furthermore, the user-optimized solution is $x_u^1 = \frac{2-a}{4}$, $x_u^2 = \frac{2+a}{4}$. Then $Z_u = F(x_u)^t x_u = -\frac{5a^3}{32} + \frac{5a^2}{16} - \frac{3a}{8} + \frac{3}{4}$. As a result,

$$\frac{Z_u}{Z_s} = -\frac{5a^3}{24} + \frac{5a^2}{12} - \frac{a}{2} + 1.$$

The bound in this case becomes $\frac{Z_u}{Z_s} \leq c^2 A^2 - 2(A - 1) = 4(1 + a^2) - 2 = 2 + 4a^2$. From this discussion we conclude that the bound is tight for all values of a satisfying

$$-\frac{5a^3}{24} + \frac{5a^2}{12} - \frac{a}{2} + 1 = 2 + 4a^2 \Leftrightarrow 5a^3 + 86a^2 + 12a + 24 = 0.$$

4 Bounds via Semidefinite Optimization

In this section, we illustrate how to bound the “price of anarchy” using ideas from semidefinite optimization. We illustrate a method to derive the same bounds as in the previous sections when the Jacobian matrix of the problem function, is not necessarily invertible, i.e., it is a positive semidefinite rather than a positive definite matrix.

Theorem 3. *For an affine variational inequality problem with problem function $F(x) = Gx + b$, with $G \succeq 0$, $b^t x \geq 0$ for all $x \in K$, the optimal objective function value of the following semidefinite optimization problem,*

$$\min \frac{a_2}{1 - a_1} \quad (2)$$

$$\text{satisfying } \begin{pmatrix} a_1 S & \frac{-G^t}{2} \\ \frac{-G}{2} & a_2 S \end{pmatrix} \succeq 0$$

$$a_2 \geq 1, \quad 0 \leq a_1 \leq 1,$$

determines the bound $\frac{Z_u}{Z_s} \leq \frac{a_2^*}{1-a_1^*}$.

Proof:

As in Theorem 1 since x_u solves the variational inequality formulation and $x_s \in K$ then

$$F(x_u)^t x_u \leq F(x_u)^t x_s = (x_u)^t G^t x_s + b^t x_s. \quad (3)$$

We wish to find $a_1, a_2 \geq 0$, so that

$$\begin{aligned} x_u^t G^t x_s &\leq a_1 x_u^t S x_u + a_2 x_s^t S x_s \quad (\text{since } S = \frac{G+G^t}{2}) \\ &= a_1 x_u^t G x_u + a_2 x_s^t G x_s. \end{aligned} \quad (4)$$

Notice that (4) is equivalent to

$$(x_u^t, x_s^t) \begin{pmatrix} a_1 G^t & \frac{-G^t}{2} \\ \frac{-G}{2} & a_2 G^t \end{pmatrix} \begin{pmatrix} x_u \\ x_s \end{pmatrix} \geq 0.$$

This relation follows when the symmetric part of the $2n \times 2n$ matrix $\begin{pmatrix} a_1 G^t & \frac{-G^t}{2} \\ \frac{-G}{2} & a_2 G^t \end{pmatrix} \succeq 0$ (that is, matrix $\begin{pmatrix} a_1 S & \frac{-G^t}{2} \\ \frac{-G}{2} & a_2 S \end{pmatrix} \succeq 0$).

Furthermore, if $a_2 \geq 1$ and $a_1 \geq 0$ then since $b^t x_u, b^t x_s \geq 0$, relations (3), (4) imply that

$$Z_u = F(x_u)^t x_u \leq a_1 F(x_u)^t x_u + a_2 F(x_s)^t x_s.$$

Therefore, if $a_1 \leq 1$ then

$$\frac{Z_u}{Z_s} \leq \frac{a_2}{1 - a_1}.$$

Given that we have freedom to select a_1 and a_2 , we find the best upper bound by solving the following semidefinite optimization problem,

$$\min \frac{a_2}{1 - a_1} \quad (5)$$

$$\text{satisfying } \begin{pmatrix} a_1 S & \frac{-G^t}{2} \\ \frac{-G}{2} & a_2 S \end{pmatrix} \succeq 0.$$

$$a_2 \geq 1, \quad 0 \leq a_1 \leq 1.$$

Notice that the optimal objective function value to the semidefinite optimization problem (5) is the bound determining the “price of anarchy”. \square

Remark: This proof does not require the matrix G to be positive definite. Matrix G in this case is positive semidefinite. The next corollary connects the bounds in Theorem 3 and 1 illustrating that when the matrix $G \succ 0$, the two bounds coincide.

Corollary 2. *For an affine variational inequality problem $F(x) = Gx + b$, with matrix $G \succ 0$, Theorems 1 and 3 yield the same bound.*

Proof:

Notice that when the matrix $G \succ 0$ (and as a result, its symmetrized matrix $S \succ 0$) then the matrix $\begin{pmatrix} a_1 S & \frac{-G^t}{2} \\ \frac{-G}{2} & a_2 S \end{pmatrix} \succeq 0$ if and only if $a_1 \geq 0$ and the matrix

$$\begin{pmatrix} a_1 \mathcal{I} & \frac{-S^{-\frac{1}{2}} G^t S^{-\frac{1}{2}}}{2} \\ \frac{-S^{-\frac{1}{2}} G S^{-\frac{1}{2}}}{2} & a_2 \mathcal{I} \end{pmatrix} \succeq 0,$$

where matrix \mathcal{I} is the identity matrix. Notice that this is also in turn equivalent to $a_1 \geq 0$ and the matrix $a_1 a_2 \mathcal{I} - \frac{S^{-\frac{1}{2}} G^t S^{-\frac{1}{2}}}{4}$ being positive semidefinite.

But this is equivalent to $a_1 \geq 0$ and $a_1 a_2 \geq \frac{\lambda_{\max}(S^{-\frac{1}{2}} G^t S^{-\frac{1}{2}})}{4} = \frac{c^2}{4}$ (see Definition 1). As a result the SDP involved in Theorem 3 yields in this case the same bound as the optimization problem (1) in Theorem 1. \square

Remark: Notice that when the constant term is zero then the SDP problem reduces to

$$\min \frac{a_2}{1 - a_1} \quad (6)$$

$$\text{satisfying } \begin{pmatrix} a_1 S & \frac{-G^t}{2} \\ \frac{-G}{2} & a_2 S \end{pmatrix} \succeq 0$$

$$a_2 \geq 0, \quad 0 \leq a_1 \leq 1.$$

Similarly to Corollary 2, one can show that when $G \succ 0$, the bound is c^2 . Notice that this is the same bound as in Corollary 1.

This approach also extends to nonlinear problems.

Theorem 4. (see [19]) For a variational inequality problem with a positive semidefinite Jacobian matrix (i.e., $\nabla F(x) \succeq 0$) satisfying the Jacobian similarity property, $F(0)^t x \geq 0$, for all $x \in K$, the optimal objective function value of the following semidefinite optimization problem,

$$\min \frac{a_2}{1 - a_1} \quad (7)$$

$$\text{satisfying } \begin{pmatrix} a_1 S(x) & \frac{-\nabla F(x)^t}{2} \\ \frac{-\nabla F(x)}{2} & a_2 S(x) \end{pmatrix} \succeq 0, \quad \forall x \in K$$

$$a_2 \geq 1, \quad 0 \leq a_1 \leq 1,$$

determines the bound $\frac{Z_u}{Z_s} \leq \frac{a_2^*}{1-a_1^*}$.

Corollary 2 extends to nonlinear problems as well.

Corollary 3. (see [19]) When the Jacobian matrix is strongly positive definite then the bound in Theorem 4 becomes the same as in Theorem 2. That is,

$$\frac{Z_u}{Z_s} \leq \begin{cases} \frac{4}{4 - c^2 A} & \text{if } c^2 \leq \frac{2}{A} \\ c^2 A^2 - 2(A - 1) & \text{if } c^2 > \frac{2}{A}. \end{cases}$$

Acknowledgments

Preparation of this paper was partly supported, in part, by the PECASE Award DMI-9984339 from the National Science Foundation, and the Singapore MIT Alliance Program.

References

1. D. Braess, “Über ein paradoxon der verkehrsplanung”, *Unternehmensforschung*, 12, 258–268, 1968.
2. C.K. Chau and K.M. Sim, “The price of anarchy for non-atomic congestion games with symmetric cost maps and elastic demands”, *Operations Research Letters*, 31, 2003.
3. R. Cole, Y. Dodis, T. Roughgarden, “Pricing Network Edges for Heterogeneous Selfish Users”, Conference version appeared in STOC, 2003.
4. J. Correa, A. Schulz, and N. Stier Moses, “Selfish Routing in Capacitated Networks”. MIT Sloan Working Paper No. 4319-03. Also to appear in the *IPCO Proceedings*, 2004.
5. S. Dafermos and F.T. Sparrow, “The traffic assignment problem for a general network”, Journal of Research of the National Bureau of Standards - B, 73B, 1969.
6. S. Dafermos, “Toll patterns for multiclass-user transportation networks”, *Transportation Science*, 211-223, 1973.

7. S. Dafermos, "Traffic equilibria and variational inequalities, *Transportation Science*, 14, 42-54, 1980.
8. P. Dubey, "Inefficiency of Nash Equilibria", *Mathematics of Operations Research*, vol. 11, 1-8, 1986.
9. F. Facchinei and J.S. Pang, "Finite-Dimensional Variational Inequalities and Complementarity Problems", Volumes I and II, Springer Verlag, Berlin, 2003.
10. M. Florian and D. Hearn, "Network Equilibrium Models and Algorithms", *Handbook of Operations Research and Management Science* , vol. 8, (M. Ball, T. Magnanti, C. Monma and G. Neumhauser eds.), Elsevier Science B.V, 1995.
11. J.H. Hammond, "Solving Asymmetric Variational Inequalities and Systems of Equations with Generalized Nonlinear Programming Algorithms", PhD Thesis, MIT, 1984.
12. D. Hearn and M. Ramana, "Solving congestion toll pricing models", *Equilibrium and Advanced Transportation Modeling*, editors P. Marcotte, S. Nguyen.
13. D. Hearn and M. Yildirim, "A toll pricing framework for traffic assignment problems with elastic demand", *Transportation and Network Analysis - Current Trends*, editors P. Marcotte, M. Gendreau.
14. R. Johari, and J.N. Tsitsiklis, "Network resource allocation and a congestion game", LIDS Working Paper, 2003.
15. E. Koutsoupias and C. H. Papadimitriou, "Worst-case equilibria," *STACS*, 1999.
16. P. Marcotte, "Network design problem with congestion effects: A case of bilevel programming" *Mathematical Programming* 34 (1986) 142-162.
17. A. Nagurney, "Network Economics; A Variational Inequality Approach" Kluwer Academic Publishers, 1993.
18. J.M. Ortega and W.C. Rheinboldt, "Iterative solution of nonlinear equations in several variables", Academic Press, New York, NY, 1970.
19. G. Perakis, "The "Price of Anarchy" when Costs are Non-separable and Asymmetric", Working Paper, ORC, MIT, submitted for publication, 2003 .
20. T. Roughgarden, "Selfish Routing", PhD Thesis, Cornell University, 2002.
21. T. Roughgarden and E. Tardos, "Bounding the Inefficiency of Equilibria in Nonatomic Congestion Games", *Journal of the ACM*, 49(2):236–259, 2002.
22. M.J. Smith, "The existence, uniqueness and stability of traffic equilibria", *Transportation Research*, 13B, 295 – 304, 1979.
23. J. Sun, "A Convergence Analysis for a Convex Version of Dikin's Algorithm", *Annals of Operations Research*, 62, 357-374, 1996.
24. J.G. Wardrop, "Some Theoretical Aspects of Road Traffic Research", in *Proceedings of the Institute of Civil Engineers*, Part II, 325-378, 1952.

Computational Complexity, Fairness, and the Price of Anarchy of the Maximum Latency Problem

Extended Abstract

José R. Correa, Andreas S. Schulz, and Nicolás E. Stier Moses

Operations Research Center, Massachusetts Institute of Technology,
77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA
`{jcorrea,schulz,nstier}@mit.edu`

Abstract. We study the problem of minimizing the maximum latency of flows in networks with congestion. We show that this problem is NP-hard, even when all arc latency functions are linear and there is a single source and sink. Still, one can prove that an optimal flow and an equilibrium flow share a desirable property in this situation: all flow-carrying paths have the same length; i.e., these solutions are “fair,” which is in general not true for the optimal flow in networks with nonlinear latency functions. In addition, the maximum latency of the Nash equilibrium, which can be computed efficiently, is within a constant factor of that of an optimal solution. That is, the so-called price of anarchy is bounded. In contrast, we present a family of instances that shows that the price of anarchy is unbounded for instances with multiple sources and a single sink, even in networks with linear latencies. Finally, we show that an s - t -flow that is optimal with respect to the average latency objective is near optimal for the maximum latency objective, and it is close to being fair. Conversely, the average latency of a flow minimizing the maximum latency is also within a constant factor of that of a flow minimizing the average latency.

1 Introduction

We study static network flow problems in which each arc possesses a latency function, which describes the common delay experienced by all flow on the arc as a function of the flow rate. Load-dependent arc costs have a variety of applications in situations in which one wants to model congestion effects, which are bound to appear, e.g., in communication networks, road traffic, or evacuation problems. In this context, a unit of flow frequently denotes a huge number of “users” (or “agents”), which might represent data packages in the Internet, drivers on a highway system, or individuals fleeing from a disaster area. Depending on the concrete circumstances, the operators of these networks can pursue a variety of system objectives. For instance, they might elect to minimize the average latency, they might aim at minimizing the maximum latency, or they might

try to ensure that users between the same origin-destination pair experience essentially the same latency. In fact, the ideal solution would be simultaneously optimal or near optimal with respect to all three objectives.

For linear latencies, we prove the existence of an s - t -flow that is at the same time optimal for two of the three objectives while its average latency is within a factor of $4/3$ of that of an optimum. As attractive as this solution might be, we also show that it is NP-hard to compute. Moreover, there is a surprising difference between linear and nonlinear latency functions. Namely, this particular flow remains optimal with respect to the maximum latency and near optimal with respect to the average latency, but it does in general not guarantee that different users face the same latency. However, an optimal s - t -flow for the average latency objective can be computed in polynomial time, and we show that the latency of any one user is within a constant factor of that of any other user. In particular, the maximum latency is within the same constant factor of the maximum latency of an optimal solution to the latter objective. This constant factor only depends on the class of allowable latency functions. For instance, its value is 2 for the case of linear latencies. Linear latencies are sufficient for certain congestion phenomena to occur. One interesting example is Braess' paradox (1968), which refers to the fact that the addition of an arc can actually increase the (average and maximum) latency in a network in which users act selfishly and independently. This user behavior is captured by the Nash equilibrium of the underlying game in which each user picks a minimal latency path, given the network congestion due to other users (Wardrop 1952). While the inefficiency of this so-called user equilibrium and hence the severity of Braess' paradox had previously been bounded in terms of the average latency, it turns out that it is also bounded with respect to the maximum latency. Indeed, the latencies encountered by different users between the same origin-destination pair are the same. The user equilibrium therefore represents another flow that can be computed in polynomial time and that is optimal or close to optimal for the three objectives introduced earlier.

The Model. We consider a directed graph $G = (N, A)$ together with a set of source-sink pairs $K \subseteq N \times N$. For each terminal pair $k = (s_k, t_k) \in K$, let \mathcal{P}_k be the set of directed (simple) paths in G from s_k to t_k , and let $d_k > 0$ be the demand rate associated with commodity k . Let $\mathcal{P} := \bigcup_{k \in K} \mathcal{P}_k$ be the set of all paths between terminal pairs, and let $d := \sum_{k \in K} d_k$ be the total demand. A feasible flow f assigns a nonnegative value f_P to every path $P \in \mathcal{P}$ such that $\sum_{P \in \mathcal{P}_k} f_P = d_k$ for all $k \in K$. In the context of single-source single-sink instances, we will drop the subindex k . Each arc a has a load-dependent latency $\ell_a(\cdot)$. We assume that the functions $\ell_a : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ are nonnegative, nondecreasing, and differentiable. We define the latency of a path $P \in \mathcal{P}$ under a given flow f as $\ell_P(f) := \sum_{a \in P} \ell_a(\sum_{Q \in \mathcal{P}: Q \ni a} f_Q)$. The maximum latency of a feasible flow f is $L(f) := \max\{\ell_P(f) : P \in \mathcal{P}, f_P > 0\}$. We call a feasible flow that minimizes the maximum latency a *min-max flow* and denote it by \hat{f} . The *maximum latency problem* consists of finding a min-max flow. The average latency of a feasible flow f is defined as $C(f) := \sum_{P \in \mathcal{P}} \ell_P(f) f_P / d$. We refer

to the optimal solution with respect to this objective function as the *system optimum* and denote it by f^* . A feasible flow is at *Nash equilibrium* (or is a *user equilibrium*) if for every $k \in K$ and every two paths $P_1, P_2 \in \mathcal{P}_k$ with $f_{P_1} > 0$, $\ell_{P_1}(f) \leq \ell_{P_2}(f)$. In other words, all flow-carrying s_k - t_k -paths have equal (and actually minimal) latency. In particular, equilibrium flows are “fair,” i.e., they have unfairness 1, if the *unfairness* of a feasible flow f is defined as $\max_{k \in K} \max\{\ell_{P_1}(f)/\ell_{P_2}(f) : P_1, P_2 \in \mathcal{P}_k, f_{P_1}, f_{P_2} > 0\}$.

Main Results. While a user equilibrium can be computed in polynomial time (Beckmann, McGuire, and Winsten 1956), and so can a system optimum if $x\ell_a(x)$ is convex for all arcs $a \in A$, we show in Section 2 that it is an NP-hard problem to compute a min-max flow. This result still holds if all latencies are linear and there is a single source-sink pair. Note that the flows that we are considering are *not* required to be integer, neither on paths nor on arcs. As pointed out earlier, a Nash equilibrium has unfairness 1 by construction. In Section 3, we establish the somewhat surprising existence of a min-max flow that is fair too, when latencies are linear and there is a single source and a single sink. In addition, although it is well known that system optima are unfair, we provide a tight bound that quantifies the severity of this effect. This bound applies to general multicommodity flows and arbitrary latency functions. Finally, in Section 4, we show that in the single-source single-sink case under arbitrary latency functions, there actually exist solutions that are simultaneously optimal or near optimal with respect to all three criteria (maximum latency, average latency, and unfairness). In fact, this property is shared by the system optimum, the user equilibrium, and—to some extent—the min-max flow, albeit with different bounds. Table 1 presents the bounds obtained for the three criteria in the single-source single-sink case with linear latencies. An important consequence of these results is that computing a user equilibrium or a system optimum constitutes a constant-factor approximation algorithm for the NP-hard maximum latency problem. On the other hand, already in networks with multiple sources and a single sink, the ratio of the maximum latency of a Nash equilibrium to that of the min-max flow is not bounded by a constant, even with linear latency functions.

Related Work. Most papers on evacuation problems consider constant travel times; we refer the reader to the surveys by Aronson (1989) and Powell, Jaillet, and Odoni (1995) for more details. One exception is the work by Köhler and Skutella (2002). They considered a dynamic quickest flow problem with load-dependent transit times, for which they established strong NP-hardness. They also provided an approximation algorithm by considering the average of a flow over time, which is a static flow. Köhler, Langkau, and Skutella (2002) proposed to use time-expanded networks to derive approximation algorithms for a similar problem.

The concept of the price of anarchy, which is the ratio of the performance of a Nash equilibrium to that of an optimal solution, was introduced by Koutsoupias and Papadimitriou (1999) in the context of

Table 1. Summary of results for single-source single-sink networks with linear latency functions. The first entry in each cell represents a worst-case bound on the ratio of the value of the flow associated with the corresponding row to the value of an optimal flow for the objective function denoted by the corresponding column. The second entry refers to the theorem in this paper in which the respective result is proved. All bounds are tight, as examples provided after each theorem demonstrate. The bound of $4/3$ on the ratio of the average latency of the user equilibrium to that of the system optimum was first proved by Roughgarden and Tardos (2002); we give a simpler proof in Theorem 7. Weitz (2001) observed first that this bound carries forward to the maximum latency objective for the case of only one source and sink; we present a generalization of this observation to multicommodity flows in Theorem 8.

	maximum latency	average latency	unfairness
min-max flow	1	$4/3$ Thm. 11	1 Thm. 5
system optimum	2 Thm. 10	1	2 Thm. 6
Nash equilibrium	$4/3$ Thm. 8	$4/3$ Thm. 7	1

a game motivated by telecommunication networks. This inspired considerable subsequent work, including Mavronicolas and Spirakis 2001; Koutsoupias, Mavronicolas, and Spirakis 2003; Czumaj and Vöcking 2002; Czumaj, Krysta, and Vöcking 2002. These papers study the maximum latency of transmissions in two-node networks consisting of multiple links connecting a single source with a single sink. Under certain assumptions, the maximum latency when users are selfish is not too large compared to the best coordinated solution. Although these results are similar in nature to some of ours, the two models are not comparable because they work with a finite number of players and consider mixed strategies. In contrast, in our setting, every player just controls an infinitesimal amount of flow, making mixed strategies essentially irrelevant. Moreover, we work with arbitrary networks. For more details on the various routing games, we refer the reader to the survey by Czumaj (2004).

Roughgarden and Tardos (2002), Roughgarden (2003), Schulz and Stier Moses (2003), and Correa, Schulz, and Stier Moses (2003) studied the price of anarchy with respect to the average travel time in general networks and for different classes of latency functions. In particular, if \mathcal{L} is the set of allowable latency functions, the ratio of the average travel time of a user equilibrium to that of a system optimum is bounded by $\alpha(\mathcal{L})$, where $\alpha(\mathcal{L})$ is a constant that only depends on \mathcal{L} . For example, in the case that \mathcal{L} only contains concave functions, $\alpha(\mathcal{L}) = 4/3$. We will later make use of this result (Section 4). For the maximum latency objective, Weitz (2001) was the first to observe that the price of anarchy is bounded in single-source single-sink networks. He also presented a family of examples that showed that Nash equilibria can be arbitrarily bad in multiple commodity networks. Roughgarden (2004) gave a tight bound for the single-source single-sink case that depends on the size of the network.

Game-theoretic concepts offer an attractive way of computing approximate solutions to certain hard problems. Indeed, Schulz and Stier Moses (2003) computed a provably good Nash equilibrium in a setting with multiple equilibria in which computing the best equilibrium is hard. Anshelevich, Deshpande, Tardos, and Wexler (2003) approximated optimal solutions to a network design problem that is NP-hard with the help of Nash and approximate Nash equilibria. A related idea was used by Fotakis et al. (2002) and Feldmann et al. (2003) to show that although it is hard to find the best and worst equilibrium of the telecommunication game described before, there exists an approximation algorithm for computing a Nash equilibrium with minimal social cost.

In the context of Section 3, we should point out that there exist multiple (nonequivalent) definitions of (un)fairness. The definition we use here comes from the competition between different agents in the routing game. Roughgarden (2002) introduced a less pessimistic version of unfairness, namely the ratio of the maximum latency of a system optimum to the latency of a user equilibrium; we later recover the bound that he obtained as a corollary to a more general result. Jahn, Möhring, Schulz, and Stier Moses (2002) considered the definition of unfairness presented here; they looked at flows that minimize the total travel time among those with bounded unfairness.

2 Computational Complexity

In our model, both the system optimum and the Nash equilibrium can be computed efficiently because they represent optimal solutions to certain convex programs.¹ On the other hand, it follows from the work of Köhler and Skutella (2002) on the quickest s - t -flow problem with load-dependent transit times that the maximum latency problem considered here is NP-hard (though not necessarily in NP) when latencies include arbitrary nonlinear functions or when there are explicit arc capacities. Lemma 1 below implies that the general maximum latency problem is in NP, while Theorem 3 establishes its NP-hardness, even in the case of linear latencies and a single source and a single sink. Note that the following result does not follow from ordinary flow decomposition as it is not clear how to convert a flow on arcs into a path flow such that the latency of the resulting paths remains bounded; in fact, it is a consequence of Theorem 3 that the latter problem is NP-hard, too.

Lemma 1. *Let f be a feasible flow for a multicommodity flow network with load-dependent arc latencies. Then there exists another feasible flow f' such that $L(f') \leq L(f)$, and f' uses at most $|A|$ paths for each source-sink pair.*

Proof. Consider an arbitrary commodity $k \in K$. Let P_1, \dots, P_r be s_k - t_k -paths such that $f_{P_i} > 0$ for $i = 1, \dots, r$, and $\sum_{i=1}^r f_{P_i} = d_k$. Slightly overloading notation, we let P_1, \dots, P_r also denote the arc incidence vectors of these paths.

¹ This statement is only true for the system optimum if we assume that $C(f)$ is convex.

Let's assume that $r > |A|$. (Otherwise we are done.) Hence, the vectors P_1, \dots, P_r are linearly dependent and $\sum_{i=1}^r \lambda_i P_i = 0$ has a nonzero solution. Let's assume without loss of generality that $\lambda_r \neq 0$. We define a new flow f'' (not necessarily feasible) by setting $f''_{P_i} := f_{P_i} - \frac{\lambda_i}{\lambda_r} f_{P_r}$ for $i = 1, \dots, r$, and $f''_P := f_P$ for all other paths P . Notice that under f'' , the flow on arcs does not change:

$$\sum_{i=1}^r P_i f''_{P_i} = \sum_{i=1}^{r-1} P_i f_{P_i} - \sum_{i=1}^{r-1} \frac{\lambda_i}{\lambda_r} P_i f_{P_r} = \sum_{i=1}^r P_i f_{P_i} .$$

Here, we used the linear dependency for the last equality. In particular, $L(f'') \leq L(f)$. Let us consider a convex combination f' of f and f'' that is nonnegative and uses fewer paths than f . Note that such a flow always exists because $f''_{P_r} = 0$, and the flow on some other paths P_1, \dots, P_{r-1} might be negative. Moreover, $L(f') \leq L(f)$, too. If f' still uses more than $|A|$ paths between s_k and t_k , we can iterate this process so long as necessary to prove the claim. \square

Corollary 2. *The decision version of the maximum latency problem is in NP.*

Proof. Lemma 1 shows the existence of a succinct certificate. Indeed, there is a min-max flow using no more than $|K| \cdot |A|$ paths. \square

We are now ready to prove that the maximum latency problem is in fact NP-hard. We present a reduction from PARTITION:

Given: A set of n positive integer numbers q_1, \dots, q_n .

Question: Is there a subset $I \subset \{1, \dots, n\}$ such that $\sum_{i \in I} q_i = \sum_{i \notin I} q_i$?

Theorem 3. *The decision version of the maximum latency problem is NP-complete, even when all latencies are linear functions and the network has a single source-sink pair.*

Proof. Given an instance of PARTITION, we define an instance of the maximum latency problem as follows. The network consists of nodes $0, 1, \dots, n$ with 0 representing the source and n the sink. The demand is one. For $i = 1, \dots, n$, the nodes $i-1$ and i are connected with two arcs, namely a_i with latency $\ell_{a_i}(x) = q_i x$ and \tilde{a}_i with latency $\ell_{\tilde{a}_i}(x) = q_i$.

Let $L := \frac{3}{4} \sum_{i=1}^n q_i$. Notice that the system optimum f^* has cost equal to L and $f_a^* = 1/2$ for all $a \in A$. We claim that the given instance of PARTITION is a YES-instance if and only if there is a solution to the maximum latency problem of maximum latency equal to L . Indeed, if there is a partition I , the flow that routes half a unit of flow along the $0-n$ -path composed of arcs a_i , $i \in I$, and \tilde{a}_i , $i \notin I$, and the other half along the complementary path has maximum latency L .

To prove the other direction, assume that we have a flow f of maximum latency equal to L . Therefore, $C(f) \leq L$ (there is unit demand), which implies that $C(f) = L$ (it cannot be better than the optimal solution). As the arc flows of a system optimum are unique, this implies that $f_a = 1/2$ for all $a \in A$. Take any

path P such that $f_P > 0$, and partition its arcs such that I contains the indices of the arcs $a_i \in P$. Then, $\frac{3}{4} \sum_{i=1}^n q_i = L = \ell_P(f) = \sum_{i \in I} \frac{q_i}{2} + \sum_{i \notin I} q_i$, and subtracting the left-hand side from the right-hand side yields $\sum_{i \in I} \frac{q_i}{4} = \sum_{i \notin I} \frac{q_i}{4}$. \square

Corollary 4. *Let f be a (path) flow in an s - t -network with linear latencies. Let $(f_a : a \in A)$ be the associated flow on arcs. Given just $(f_a : a \in A)$ and $L(f)$, it is NP-hard to compute a decomposition of this arc flow into a (path) flow f' such that $L(f') \leq L(f)$. In particular, it is NP-hard to recover a min-max flow even though its arc values are given.*

Note that Corollary 4 neither holds for the system optimum nor the user equilibrium. Any flow derived from an ordinary flow decomposition is indeed an optimal flow or an equilibrium flow, respectively. Let us also mention that Theorem 4.3 in Köhler and Skutella (2002) implies that the maximum latency problem is APX-hard when latencies can be arbitrary nonlinear functions.

3 Fairness

User equilibria are fair by definition. Indeed, all flow-carrying paths between the same source and sink have equal latency. The next result establishes the same property for min-max s - t -flows in the case of linear latencies. Namely, a fair min-max flow always exists. Therefore, the difference between a Nash equilibrium and a min-max flow is that the latter may leave paths unused that are shorter than the ones carrying flow, a situation that cannot happen in equilibrium. The following result is not true for nonlinear latencies, as we shall see later.

Theorem 5. *Every instance of the single-source single-sink maximum latency problem with linear latency functions has an optimal solution that is fair.*

Proof. Consider an instance with demand d and latency functions $\ell_a(f_a) = q_a f_a + r_a$, for $a \in A$. Among all min-max flows, let \hat{f} be the one that uses the smallest number of paths. Let P_1, P_2, \dots, P_k be these paths. Consider the following linear program:

$$\min z \tag{1}$$

$$\text{s.t. } \sum_{a \in P_i} \left(q_a \left(\sum_{P_h \ni a} f_{P_h} \right) + r_a \right) \leq z \quad \text{for } i = 1, \dots, k, \tag{2}$$

$$\sum_{i=1}^k f_{P_i} = d \tag{3}$$

$$f_{P_i} \geq 0 \quad \text{for } i = 1, \dots, k. \tag{4}$$

Note that this linear program has $k+1$ variables. Furthermore, by construction, it has a feasible solution with $z = L(\hat{f})$, and there is no solution with $z < L(\hat{f})$.

Therefore, an optimal basic feasible solution gives a min-max flow that satisfies with equality k of the inequalities (2) and (4). As $f_{P_i} > 0$ for all i because of the minimality assumption, all inequalities (2) have to be tight. \square

A byproduct of this proof is that an arbitrary flow can be transformed into a fair one without increasing its maximum latency. In fact, just solve the corresponding linear program. An optimal basic feasible solution will either be fair or it will use fewer paths. In the latter case, eliminate all paths carrying zero flow and repeat until a fair solution is found.

Notice that the min-max flow may not be fair for nonlinear functions. Indeed, the instance displayed in Figure 1 features high unfairness with latencies that are polynomials of degree $p \geq 2$. When $a = (1 + \varepsilon)^{p-1}$ and $b = 2 - (\frac{1+\varepsilon}{2+\varepsilon})^{p-1} - \delta$ for some $\varepsilon > 0$ and $\delta > 0$ such that $b > 1$, the min-max flow routes $\frac{1}{2+\varepsilon}$ units of flow along the “top-bottom” and “bottom-top” paths, respectively, and $\frac{\varepsilon}{2+\varepsilon}$ units of flow along the “top-top” path. It is not hard to see that this flow is optimal. Indeed, the “bottom-bottom” path is too long to carry any flow. Moreover, by symmetry, the “top-bottom” and “bottom-top” paths have to carry the same amount of flow. Therefore, the optimal solution can be computed by solving a one-dimensional minimization problem, whose only variable is the amount x of flow on the “top-top” path. The unique optimal solution to this problem is $x = \frac{\varepsilon}{2+\varepsilon}$. Let us compute the unfairness of this solution. The “top-top” path has latency equal to $2(\frac{1+\varepsilon}{2+\varepsilon})^p$, which tends to $(\frac{1}{2})^{p-1}$ as $\varepsilon \rightarrow 0$. The latency of the other two paths used by the optimum is equal to $2 - \delta$. Therefore, the unfairness of this min-max flow is arbitrarily close to 2^p .

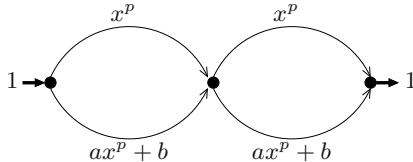


Fig. 1. Fair min-max flows may not exist for instances with nonlinear latencies.

A typical argument against using the system optimum in the design of route-guidance devices for traffic assignment is that, in general, it assigns some drivers to unacceptably long paths in order to use shorter paths for most other drivers; see, e.g., Beccaria and Bolelli (1992). The following theorem quantifies the severity of this effect by characterizing the unfairness of the system optimum. It turns out that there is a relation to earlier work by Roughgarden (2002), who compared the maximum latency of a system optimum in a single-sink single-source network to the latency of a user equilibrium. He showed that for a given class of latency functions \mathcal{L} , this ratio is bounded from above by $\gamma(\mathcal{L})$, which is defined to be the smallest value that satisfies $\ell_a^*(x) \leq \gamma(\mathcal{L})\ell_a(x)$ for all $\ell \in \mathcal{L}$ and all $x \geq 0$. Here, $\ell_a^*(x) := \ell_a(x) + x\ell'_a(x)$ is the function that makes a system

optimum for the original instance a user equilibrium of an instance in which the latencies are replaced by ℓ^* (Beckmann, McGuire, and Winsten 1956). For instance, $\gamma(\text{polynomials of degree } p) = p + 1$. We prove that the unfairness of a system optimum is in fact bounded by the same constant, even for general instances with multiple commodities. The same result was independently obtained by Roughgarden (personal communication, October 2003).

Theorem 6. *Let f^* be a system optimum in a multicommodity flow network with arc latency functions drawn from a class \mathcal{L} . Then, the unfairness of f^* is bounded from above by $\gamma(\mathcal{L})$.*

Proof. We will prove the result for the single-source single-sink case. The extension to the general case is straightforward. As a system optimum is a user equilibrium with respect to latencies ℓ^* , there exists L^* such that $\ell_P^*(f^*) = L^*$ for all paths $P \in \mathcal{P}$ with $f_P^* > 0$. From the definitions of ℓ^* and $\gamma(\mathcal{L})$, we have that $\ell_a(x) \leq \ell_a^*(x) \leq \gamma(\mathcal{L})\ell_a(x)$ for all x . Let $P_1, P_2 \in \mathcal{P}$ be two arbitrary paths with $f_{P_1}^*, f_{P_2}^* > 0$. Hence, $\ell_{P_1}(f^*) \leq L^*$ and $\ell_{P_2}(f^*) \geq L^*/\gamma(L)$. It follows that $\ell_{P_1}(f^*)/\ell_{P_2}(f^*) \leq \gamma(L)$. \square

Notice that Theorem 6 implies Roughgarden's earlier bound for the single-source single-sink case. Indeed, for a Nash equilibrium f , $\min\{\ell_P(f^*) : P \in \mathcal{P}, f_P^* > 0\} \leq \min\{\ell_P(f) : P \in \mathcal{P}, f_P > 0\}$. Otherwise, $C(f^*) > C(f)$, which contradicts the optimality of f^* . In addition, the example shown in Figure 2 proves that the bound given in Theorem 6 is tight. Indeed, it is easy to see that the system optimum routes half of the demand along each arc, implying that the unfairness is $\ell^*(d/2)/\ell(d/2)$. Taking the supremum of that ratio over $d \geq 0$ and $\ell \in \mathcal{L}$ yields $\gamma(\mathcal{L})$.

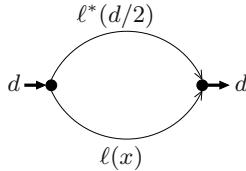


Fig. 2. Instance showing that Theorem 6 is tight.

4 Price of Anarchy and Related Approximation Results

Nash equilibria in general and user equilibria in particular are known to be inefficient, as evidenced by Braess' paradox (1968). Koutsoupias and Papadimitriou (1999) suggested measuring this degradation in performance, which results from the lack of central coordination, by the worst-case ratio of the value of an equilibrium to that of an optimum. This ratio has become known as the “price of anarchy,” a phrase coined by

Papadimitriou (2001). It is quite appealing (especially for evacuation situations) that in the routing game considered here, the price of anarchy is small; i.e., the selfishness of users actually drives the solution close to optimality. Recall that the user equilibrium results from everyone choosing a shortest path under the prevailing congestion conditions. Since a user equilibrium can be computed in polynomial time, this also leads to an approximation algorithm for the maximum latency problem.

In order to derive a bound on the price of anarchy for the maximum latency objective, we use a corresponding bound for the average latency of Nash equilibria, which was first proved for linear latency functions by Roughgarden and Tardos (2002), and then extended to different classes of latency functions by Roughgarden (2003), Schulz and Stier Moses (2003), and Correa, Schulz, and Stier Moses (2003). For the sake of completeness, let us include a simpler proof of Roughgarden and Tardos' result (see also Correa, Schulz, and Stier Moses 2003).

Theorem 7 (Roughgarden and Tardos 2002). *Let f be a user equilibrium and let f^* be a system optimum in a multicommodity flow network with linear latency functions. Then $C(f) \leq \frac{4}{3}C(f^*)$.*

Proof. Let $\ell_a(x) = q_a x + r_a$ with $q_a, r_a \geq 0$ for all $a \in A$. W.l.o.g., $d = 1$. Then,

$$\begin{aligned} C(f) &= \sum_{a \in A} (q_a f_a + r_a) f_a \leq \sum_{a \in A} (q_a f_a + r_a) f_a^* \leq \\ &\quad \sum_{a \in A} (q_a f_a^* + r_a) f_a^* + \frac{1}{4} \sum_{a \in A} q_a f_a^2 \leq C(f^*) + \frac{1}{4} C(f) . \end{aligned}$$

The first inequality holds since the equilibrium flow f uses shortest paths with respect to the arc latencies caused by itself. The second inequality follows from $(f_a^* - f_a/2)^2 \geq 0$. \square

For general latency functions $\ell \in \mathcal{L}$,

$$C(f) \leq \alpha(\mathcal{L}) C(f^*), \text{ where } \alpha(\mathcal{L}) := \left(1 - \sup_{\ell \in \mathcal{L}, 0 \leq x \leq d} \left\{ \frac{x(\ell(d) - \ell(x))}{d \ell(d)} \right\}\right)^{-1} , \quad (5)$$

and the proof is similar to the one given above for Theorem 7; see Roughgarden (2003) and Correa, Schulz, and Stier Moses (2003) for details. For polynomials with nonnegative coefficients of degree 2, $\alpha(\mathcal{L})$ equals 1.626; for those with degree 3, $\alpha(\mathcal{L}) = 1.896$; in general, $\alpha(\mathcal{L}) = \Theta(p/\ln p)$ for polynomials of degree p . It was first noted by Weitz (2001) that in networks with only one source and one sink, any upper bound on the price of anarchy for the average latency is an upper bound on the price of anarchy for the maximum latency. We include a multicommodity version of this result.

Theorem 8. *Consider a multicommodity flow network with latency functions in \mathcal{L} . Let f be a Nash equilibrium and \hat{f} a min-max flow. For each commodity $k \in K$, $L_k(f) \leq \frac{d}{d_k} \alpha(\mathcal{L}) L(\hat{f})$, where L_k is the maximum latency incurred by commodity k , d_k is its demand rate, and d is the total demand.*

Proof. Let f^* be the system optimum. Then,

$$d_k L_k(f) \leq d C(f) \leq d \alpha(\mathcal{L}) C(f^*) \leq d \alpha(\mathcal{L}) C(\hat{f}) \leq d \alpha(\mathcal{L}) L(\hat{f}) .$$

Here, the first inequality holds because f is a Nash equilibrium, the second inequality is exactly (5), the third one comes from the optimality of f^* , and the last one just says that the average latency is less than the maximum latency. \square

The proof of Theorem 8 implies that if for a given single-source single-sink instance an equilibrium flow f happens to be a system optimum, then f is also optimal for the maximum latency objective.² For single-source single-sink networks with general latency functions, Theorem 8 shows that computing a Nash equilibrium is an $\alpha(\mathcal{L})$ -approximation algorithm for the maximum latency problem. Notice that this guarantee is tight as shown by the example given in Figure 3, which goes back to Braess (1968). Indeed, the latency of a Nash equilibrium is $\ell(d)$ while the maximum latency of a min-max flow, which coincides with the system optimum, is $\ell(d) - \max_{0 \leq x \leq d} \left\{ \frac{x}{d} (\ell(d) - \ell(x)) \right\}$. Taking the supremum over $d \geq 0$ and $\ell \in \mathcal{L}$, the ratio of the latency of the Nash equilibrium to that of the min-max flow is arbitrarily close to $\alpha(\mathcal{L})$.

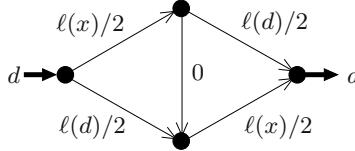


Fig. 3. Instance showing that Theorem 8 is tight for single-commodity networks.

For instances with multiple sources and a single sink, the maximum latency of a user equilibrium is unbounded with respect to that of an optimal solution, even with linear latencies. In fact, we will show that the price of anarchy cannot be better than $\Omega(n)$, where n is the number of nodes in the network. Note that this also implies that the price of anarchy is unbounded in single-source single-sink networks with explicit arc capacities. Weitz (2001) showed that the price of anarchy is unbounded in the case of two commodities, and Roughgarden (2004) proved that it is at most $n - 1$ if there is a common source and sink.

Theorem 9. *The price of anarchy in a single-commodity network with multiple sources and a single sink is $\Omega(n)$, even if all latencies are linear functions.*

Proof. Fix a constant $\varepsilon > 0$ and consider the instance presented in Figure 4. Nodes $n, n-1, \dots, 1$ are the sources while node 0 is the sink. Nodes i and $i-1$ are

² For instance, if all latency functions are monomials of the same degree but with arc-dependent coefficients, it is well known that user equilibria and system optima coincide (Dafermos and Sparrow 1969).

connected with two arcs: a_i with constant latency equal to 1 and \tilde{a}_i with latency equal to x/ε^i . Let the demand entering node $i > 0$ be ε^i . The user equilibrium of this instance routes the flow along paths of the form $\tilde{a}_i, a_{i-1}, \dots, a_1$ and has maximum latency n . To show the claim, it suffices to exhibit a good solution. For instance, for origin i , let its demand flow along the path $a_i, \tilde{a}_{i-1}, \dots, \tilde{a}_1$. Under this flow, the load of \tilde{a}_i is equal to $\varepsilon^{i+1} + \dots + \varepsilon^n$ and its traversal time is $(\varepsilon^{i+1} + \dots + \varepsilon^n)/\varepsilon^i = \varepsilon^1 + \dots + \varepsilon^{n-i}$. Hence, we can bound the maximum latency from above by $1 + \frac{n\varepsilon}{1-\varepsilon}$, which tends to 1 when $\varepsilon \rightarrow 0$. \square

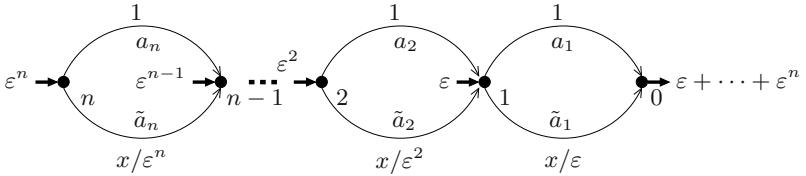


Fig. 4. Nash equilibria can be arbitrarily bad when multiple sources are present.

In the single-source single-sink case, Nash equilibria are not the only good approximations to the maximum latency problem; an immediate corollary of Theorem 6 is that system optima are also close to optimality with respect to the maximum latency objective.

Theorem 10. *For single-source single-sink instances with latency functions drawn from \mathcal{L} , computing a system optimum is a $\gamma(\mathcal{L})$ -approximation algorithm for the maximum latency problem.*

Proof. Theorem 6 states that the length of a longest path used by the system optimum f^* is at most $\gamma(\mathcal{L})$ times the length of a shortest flow-carrying path. The latter value cannot be bigger than the maximum latency of a path used by the min-max flow because f^* is optimal for the average latency; the result follows. \square

The bound given in Theorem 10 is best possible. To see this, consider the instance depicted in Figure 5. The min-max flow routes the entire demand along the lower arc, for a small enough $\varepsilon > 0$. On the other hand, the unique system optimum has to satisfy $\ell^*(f^*) = \ell^*(d) - \varepsilon$, where f^* is the flow along the lower arc. Therefore, the upper arc has positive flow and the maximum latency is $\ell^*(d) - \varepsilon$. The ratio between the maximum latencies of the two solutions is arbitrarily close to $\ell^*(d)/\ell(d)$. Taking the supremum over $d \geq 0$ and $\ell \in \mathcal{L}$ shows that the bound in Theorem 10 is tight.

To complete Table 1, let us prove that the average latency of the min-max flow is not too far from that of the system optimum.

Theorem 11. *Let \hat{f} be a min-max flow and let f^* be a system optimum for an instance with a single source, a single sink and latencies drawn from \mathcal{L} . Then, $C(\hat{f}) \leq \alpha(\mathcal{L})C(f^*)$.*

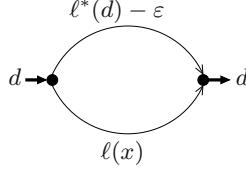


Fig. 5. Instance showing that Theorem 10 is tight.

Proof. Note that $C(\hat{f}) \leq L(\hat{f}) \leq L(f) = C(f) \leq \alpha(\mathcal{L})C(f^*)$, where f is the Nash equilibrium of the instance. \square

Again, the guarantee given in the previous theorem is tight. To show this, it is enough to note that the equilibrium flow and the min-max flow coincide in the example of Figure 6, and their average latency is $\ell(d)$. Moreover, the average latency of the system optimum is arbitrary close to $\ell(d) - \max_{0 \leq x \leq d} \left\{ \frac{x}{d} (\ell(d) - \ell(x)) \right\}$. Taking the supremum of the ratio of these two values over $d \geq 0$ and $\ell \in \mathcal{L}$ completes the argument.

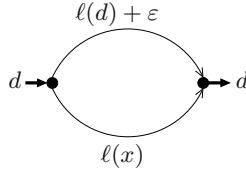


Fig. 6. Instance showing that Theorem 11 is tight.

5 Concluding Remarks

While Table 2 summarizes our findings for single-source single-sink networks with latencies drawn from a given class \mathcal{L} of allowable latency functions, let us briefly discuss another useful objective function. A different way of interpreting the two-node scenario of Koutsoupias and Papadimitriou (1999) in which each path consists of a single arc, is to minimize the maximum latency over flow-carrying arcs. In contrast to the maximum latency problem, this problem can be solved in polynomial time when latency functions are convex. However, the optimal “bottleneck flow” can be arbitrarily bad with respect to all three objectives considered in the main part of this paper. On the other hand, the equilibrium flow, the system-optimal flow, and the min-max flow can all at once be arbitrarily bad with respect to the bottleneck objective. We leave the details to the full version of this paper.

Table 2. Overview of approximation guarantees for single-source single-sink networks when latencies belong to a given set \mathcal{L} . All bounds are tight. The “?” indicates that no upper bound is known; recall from the example depicted in Figure 1 that 2^p is a lower bound for polynomials of degree $p \geq 2$.

	maximum latency	average latency	unfairness
min-max flow	1	$\alpha(\mathcal{L})$?
system optimum	$\gamma(\mathcal{L})$	1	$\gamma(\mathcal{L})$
user equilibrium	$\alpha(\mathcal{L})$	$\alpha(\mathcal{L})$	1

References

- Anshelevich, E., A. Desgupta, É. Tardos, and T. Wexler (2003). Near-optimal network design with selfish agents. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing* (STOC), San Diego, CA, pp. 511–520. ACM Press, New York, NY.
- Aronson, J. E. (1989). A survey of dynamic network flows. *Annals of Operations Research* 20, 1–66.
- G. Beccaria and A. Bolelli (1992). Modelling and assessment of dynamic route guidance: The MARGOT project. In *Proceedings of the IEEE Vehicle Navigation & Information Systems Conference*, Oslo, Norway, pp. 117–126.
- Beckmann, M. J., C. B. McGuire, and C. B. Winsten (1956). *Studies in the Economics of Transportation*. Yale University Press, New Haven, CT.
- Braess, D. (1968). Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung* 12, 258–268.
- Correa, J. R., A. S. Schulz, and N. E. Stier Moses (2003). Selfish routing in capacitated networks. MIT, Sloan School of Management, Working Paper No. 4319-03. To appear in *Mathematics of Operations Research*.
- Czumaj, A. (2004). Selfish routing on the Internet. In J. Leung (Ed.), *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Boca Raton, FL. To appear.
- Czumaj, A., P. Krysta, and B. Vöcking (2002). Selfish traffic allocation for server farms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing* (STOC), Montreal, Canada, pp. 287–296. ACM Press, New York, NY.
- Czumaj, A. and B. Vöcking (2002). Tight bounds for worst-case equilibria. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), San Francisco, CA, pp. 413–420. SIAM, Philadelphia, PA.
- Dafermos, S. C. and F. T. Sparrow (1969). The traffic assignment problem for a general network. *Journal of Research of the U.S. National Bureau of Standards* 73B, 91–118.
- Feldmann, R., M. Gairing, T. Lücking, B. Monien, and M. Rode (2003). Nashification and the coordination ratio for a selfish routing game. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger (Eds.), *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming* (ICALP), Eindhoven, The Netherlands, Volume 2719 of *Lecture Notes in Computer Science*, pp. 514–526. Springer, Berlin.

- Fotakis, D., S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis (2002). The structure and complexity of Nash equilibria for a selfish routing game. In P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo (Eds.), *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP)*, Málaga, Spain, Volume 2380 of *Lecture Notes in Computer Science*, pp. 123–134. Springer, Berlin.
- Jahn, O., R. H. Möhring, A. S. Schulz, and N. E. Stier Moses (2002). System-optimal routing of traffic flows with user constraints in networks with congestion. MIT, Sloan School of Management, Working Paper No. 4394-02.
- Köhler, E., K. Langkau, and M. Skutella (2002). Time-expanded graphs with flow-dependent transit times. In R. H. Möhring and R. Raman (Eds.), *Proceedings of the 10th Annual European Symposium on Algorithms (ESA)*, Rome, Italy, Volume 2461 of *Lecture Notes in Computer Science*, pp. 599–611. Springer, Berlin.
- Köhler, E. and M. Skutella (2002). Flows over time with load-dependent transit times. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, San Francisco, CA, pp. 174–183. SIAM, Philadelphia, PA.
- Koutsoupias, E., M. Mavronicolas, and P. Spirakis (2003). Approximate equilibria and ball fusion. *Theory of Computing Systems* 36, 683–693.
- Koutsoupias, E. and C. H. Papadimitriou (1999). Worst-case equilibria. In C. Meinel and S. Tison (Eds.), *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, Trier, Germany, Volume 1563 of *Lecture Notes in Computer Science*, pp. 404–413. Springer, Berlin.
- Mavronicolas, M. and P. Spirakis (2001). The price of selfish routing. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, Hersonissos, Greece, pp. 510–519. ACM Press, New York, NY.
- Papadimitriou, C. H. (2001). Algorithms, games, and the Internet. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, Hersonissos, Greece, pp. 749–753. ACM Press, New York, NY.
- Powell, W. B., P. Jaillet, and A. Odoni (1995). Stochastic and dynamic networks and routing. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser (Eds.), *Networks*, Volume 4 of *Handbook in Operations Research and Management Science*, pp. 141–295. Elsevier Science, Amsterdam.
- Roughgarden, T. (2002). How unfair is optimal routing? In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, San Francisco, CA, pp. 203–204. SIAM, Philadelphia, PA.
- Roughgarden, T. (2003). The price of anarchy is independent of the network topology. *Journal of Computer and System Sciences* 67, 341–364.
- Roughgarden, T. (2004). The maximum latency of selfish routing. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, New Orleans, LA, pp. 973–974. SIAM, Philadelphia, PA.
- Roughgarden, T. and É. Tardos (2002). How bad is selfish routing? *Journal of the ACM* 49, 236–259.
- Schulz, A. S. and N. E. Stier Moses (2003). On the performance of user equilibria in traffic networks. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Baltimore, MD, pp. 86–87. SIAM, Philadelphia, PA.
- Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers* 1, Part II, 325–378.
- Weitz, D. (2001). The price of anarchy. Unpublished manuscript.

Polynomial Time Algorithm for Determining Optimal Strategies in Cyclic Games

Dmitrii Lozovanu

Institute of Mathematics and Computer Science,
Academy of Sciences, Academy str., 5, Kishinev, MD–2028, Moldova
lozovanu@math.md

Abstract. The problem of finding the value and optimal strategies of players in cyclic games is studied. A polynomial time algorithm for solving cyclic games is proposed.

1 Introduction and Problem Formulation

Let $G = (V, E)$ be a finite directed graph in which every vertex $u \in V$ has at least one leaving edge $e = (u, v) \in E$. On edge set E is given a function $c: E \rightarrow R$ which assign a cost $c(e)$ to each edge $e \in E$. In addition the vertex set V is divided into two disjoint subsets V_A and V_B ($V = V_A \cup V_B$, $V_A \cap V_B = \emptyset$) which we will regard as a positions sets of two players.

On G we consider the following two-person game from [1,2]. The game start at position $v_0 \in V$. If $v_0 \in V_A$ then the move is done by first player, otherwise it is done by second one. The move means the passage from position v_0 to the neighbour position v_1 through the edge $e_1 = (v_0, v_1) \in E$. After that if $v_1 \in V_A$ then the move is done by first player, otherwise it is done by second one and so on indefinitely. The first player has the aim to maximize $\liminf_{t \rightarrow \infty} \frac{1}{t} \sum_{i=1}^t c(e_i)$

while the second player has the aim to minimize $\limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{i=1}^t c(e_i)$.

In [1,2] is proved that for this game there exists a value $p(v_0)$ such that the first player has a strategy of moves that insures $\liminf_{t \rightarrow \infty} \frac{1}{t} \sum_{i=1}^t c(e_i) \geq p(v_0)$ and the second player has a strategy of moves that insures $\limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{i=1}^t c(e_i) \leq p(v_0)$. Furthermore in [1,2] is shown that the players can achieve the value $p(v_0)$ applying the strategies of moves which do not depend on t . This means that the considered game can be formulated in the terms of stationary strategies. Such statement of the game in [2] is named cyclic game.

The strategies of players in cyclic game are defined as a maps

$$s_A: u \rightarrow v \in V_G(u) \quad \text{for } u \in V_A; \quad s_B: u \rightarrow v \in V_G(u) \quad \text{for } u \in V_B,$$

where $V_G(u)$ represents the set of extremities of edges $e = (u, v) \in E$, i.e. $V_G(u) = \{v \in V \mid e = (u, v) \in E\}$. Since G is a finite graph then the sets of strategies of players

$$S_A = \{s_A: u \rightarrow v \in V_G(u) \text{ for } u \in V_A\}; S_B = \{s_B: u \rightarrow v \in V_G(u) \text{ for } u \in V_B\}$$

are finite sets. The payoff function $F_{v_0}: S_A \times S_B \rightarrow R$ in cyclic game is defined as follows.

Let $s_A \in S_A$ and $s_B \in S_B$ be a fixed strategies of players. Denote by $G_s = (V, E_s)$ the subgraph of G generated by edges of form $(u, s_A(u))$ for $u \in V_A$ and $(u, s_B(u))$ for $u \in V_B$. Then G_s contains a unique directed cycle C_s which can be reached from v_0 through the edges $e \in E_s$. The value $F_{v_0}(s_A, s_B)$ we consider equal to mean edges cost of cycle C_s , i.e.

$$F_{v_0}(s_A, s_B) = \frac{1}{n(C_s)} \sum_{e \in E(C_s)} c(e),$$

where $E(C_s)$ represents the set of edges of cycle C_s and $n(C_s)$ is a number of the edges of C_s . So, the cyclic games is determined uniquely by the network (G, V_A, V_B, c) and starting position v_0 . In [1,2] is proved that there exist the strategies $s_A^* \in S_A$ and $s_B^* \in S_B$ such that

$$p(v) = F_v(s_A^*, s_B^*) = \max_{s_A \in S_A} \min_{s_B \in S_B} F_v(s_A, s_B) = \min_{s_B \in S_B} \max_{s_A \in S_A} F_v(s_A, s_B), \forall v \in V.$$

So, the optimal strategies s_A^*, s_B^* of players in cyclic games do not depend on starting position v although for different positions $u, v \in V$ the values $p(u)$ and $p(v)$ may be different. It means that the positions set V can be divided into several classes $V = V^1 \cup V^2 \cup \dots \cup V^k$ according to values of positions p^1, p^2, \dots, p^k , i.e. $u, v \in V^i$ if and only if $p^i = p(u) = p(v)$. In the case $k = 1$ the network (G, V_A, V_B, c) is named the ergodic network [2]. In [3] is shown that every cyclic game with arbitrary network (G, V_A, V_B, c) and given starting position v_0 can be reduced to an auxiliary cyclic game on auxiliary ergodic network (G', V'_A, V'_B, c') .

It is well-known [2,4] that the decision problem associated to cyclic game is in $NP \cap co-NP$ but it is not yet known for this problem to be in P . Some exponential and pseudo-exponential algorithms for finding the value and the optimal strategies of players in cyclic are proposed in [1,2,4].

A similar problems on acyclic networks have been considered in [5,6]. A new classes of acyclic games have been studied in [5,6] and polynomial time algorithms for its solving have been elaborated. Moreover in [5,6] is made an attempt to reduce the cyclic game to acyclic one. Unfortunately the reduction from [5,6] gives the first player some control over the length of the formed cycles, so that it is not conservative in the general case.

In this paper we propose a polynomial time algorithm for finding the value and optimal strategies of players in cyclic games. The proposed algorithm is based on a new reduction from cyclic games to acyclic one.

2 Some Preliminary Results

First of all we need to remind some preliminary results from [2].

Let (G, V_A, V_B, c) be a network with the properties described in section 1. We denote

$$\text{ext}(c, u) = \begin{cases} \max_{v \in V_G(u)} \{c(u, v)\} & \text{for } u \in V_A, \\ \min_{v \in V_G(u)} \{c(u, v)\} & \text{for } u \in V_B, \end{cases}$$

$$VEXT(c, u) = \{v \in V_G(u) \mid c(u, v) = \text{ext}(c, u)\}.$$

We shall use the potential transformation $c'(u, v) = c(u, v) + \varepsilon(v) - \varepsilon(u)$ for costs on edges $e = (u, v) \in E$, where $\varepsilon: V \rightarrow R$ is an arbitrary function on vertex set V . In [2] is noted that the potential transformation do not change the value and the optimal strategies of players in cyclic games.

Theorem 1. *Let (G, V_A, V_B, c) be an arbitrary network with the properties described in section 1. Then there exists the value $p(v)$, $v \in V$ and the function $\varepsilon: V \rightarrow R$ which determine a potential transformation $c'(u, v) = c(u, v) + \varepsilon(v) - \varepsilon(u)$ for costs on edges $e = (u, v) \in E$, such that the following properties holds*

- a) $p(u) = \text{ext}(c', u)$ for $v \in V$,
- b) $p(u) = p(v)$ for $u \in V_A \cup V_B$ and $v \in VEXT(c', u)$,
- c) $p(u) \geq p(v)$ for $u \in V_A$ and $v \in V_G(u)$,
- d) $p(u) \leq p(v)$ for $u \in V_B$ and $v \in V_G(u)$,
- e) $\max_{e \in E} |c'(e)| \leq 2|V| \max_{e \in E} |c(e)|$.

The values $p(v)$, $v \in V$ on network (G, V_A, V_B, c) are determined unequally and the optimal strategies of players can be found in the following way: fix the arbitrary strategies $s_A^*: V_A \rightarrow V$ and $s_B^*: V_B \rightarrow V$ such that $s_A^*(u) \in VEXT(c', u)$ for $u \in V_A$ and $s_B^*(u) \in VEXT(c', u)$ for $u \in V_B$.

The proof of Theorem 1 is given in [2].

Further we shall use the theorem 1 in the case of the ergodic network (G, V_1, V_2, c) , i.e. we shall use the following corollary.

Corollary 2. *Let (G, V_A, V_B, c) be an ergodic network. Then there exists the value p and the function $\varepsilon: V \rightarrow R$ which determine a potential transformation $c'(u, v) = c(u, v) + \varepsilon(v) - \varepsilon(u)$ for costs of edges $e = (u, v) \in E$ such that $p = \text{ext}(c', u)$ for $u \in V$. The optimal strategies of players can be found as follows: fix arbitrary strategies $s_A^*: V_A \rightarrow V$ and $s_B^*: V_B \rightarrow V$ such that $s_A^*(u) \in VEXT(c', u)$ for $u \in V_A$ and $s_B^*(u) \in VEXT(c', u)$ for $u \in V_B$.*

3 The Reduction of Cyclic Games to Ergodic Ones

Let us consider an arbitrary network (G, V_A, V_B, c) with given starting position $v_0 \in V$ which determine a cyclic game. In [3] is shown that this game can be reduced to a cyclic game on auxiliary ergodic network (G', W_A, W_B, \bar{c}) , $G' = (W, F)$ in which is preserving the value $p(v_0)$ and $v_0 \in W = V \cup X \cup Y$.

The graph $G' = (W, F)$ is obtained from G if each edge $e = (u, v)$ is changed by a triple of edges $e^1 = (u, x), e^2 = (x, y), e^3 = (y, v)$ with the costs $\bar{c}(e^1) = \bar{c}(e^2) = \bar{c}(e^3) = c(e)$. Here $x \in X, y \in Y$ and $u, v \in V; W = V \cup X \cup Y$. In addition in G' each vertex x is connected with v_0 by edge (x, v_0) with the cost $\bar{c}(x, v_0) = M$ (M is a great value) and each edge (y, v_0) is connected with v_0 by edge (y, v_0) with the cost $\bar{c}(y, v_0) = -M$. In (G', W_A, W_B, \bar{c}) the sets W_A and W_B are defined as follows: $W_A = V_A \cup Y; W_B = V_B \cup X$.

It is easy to observe that this reduction can be done in linear time.

4 Acyclic Games on Networks and Polynomial Algorithms for Its Solving

We consider the following auxiliary games from [5,6].

4.1 Acyclic c -Game on Acyclic Networks

Let $G = (V, E)$ be a finite directed graph without directed cycles. Assume that in G there exists a vertex v_f which is attainable from each vertex $v \in V$, i.e. v_f is a sink in G . In addition we consider that on edge set E is given a function $c: E \rightarrow R$ and the vertex set V is divided into two disjoint subsets V_A and V_B ($V = V_A \cup V_B, V_A \cap V_B = \emptyset$) which we regard as positions sets of two players.

On G we consider the two-person game from [5]. The game starts at given position $v_0 \in V \setminus \{v_f\}$. If $v_0 \in V_A$ then the move is done by first player otherwise it is done by second one. Like in cyclic game here the move means the passage from position v_0 to the neighbor position v_1 through the edge $e_1 = (v_0, v_1)$. Again if $v_A \in V_A$ then the move is done by first player, otherwise it is done by second one and so on while the final position v_f is not reached. As soon as the final position v_f is reached the game is over. The final step of the game we denote by t . In this game the first player has the aim to maximize the integral cost $\sum_{i=1}^t c(e_i)$ and the second player has the aim to minimize $\sum_{i=1}^t c(e_i)$.

In [5] is shown that for such game there exists a value $\hat{p}(v_0)$ such that the first player has a strategy of moves that insures $\sum_{i=1}^t c(e_i) \geq \hat{p}(v_0)$ and the second player has a strategy of moves that insures $\sum_{i=1}^t c(e_i) \leq \hat{p}(v_0)$. Moreover the players can active this values using the stationary strategies of moves. This game in [5] is named acyclic c -game. The strategies of players in acyclic c -game can be defined as a maps

$$s_A: u \rightarrow v \in V_G(u) \text{ for } u \in V_A \setminus \{v_0\}; s_B: u \rightarrow v \in V_G(u) \text{ for } u \in V_B \setminus \{v_0\}.$$

Since G is a finite graph then the set of strategies of players

$$S_A = \{s_A: u \rightarrow v \in V_G(u) \text{ for } u \in V_A \setminus \{v_0\}\};$$

$$S_B = \{s_B: u \rightarrow v \in V_G(u) \text{ for } u \in V_B \setminus \{v_0\}\}$$

are finite sets. The payoff function $\hat{F}_{v_0}: S_A \times S_B \rightarrow R$ in acyclic c -game is defined as follows.

Let $s_A \in S_A$ and $s_B \in S_B$ be a fixed strategies of players. Denote by $T_s = (V, E_s)$ the tree with root vertex v_f in G generated by edges of form $(u, s_A(u))$ for $u \in V_A \setminus \{v_0\}$ and $(u, s_B(u))$ for $u \in V_B \setminus \{v_0\}$. Then in T_s there exists a unique directed path $P_{T_s}(v_0, v_f)$ from v_0 to v_f . We put

$$\hat{F}_{v_0}(s_A, s_B) = \sum_{e \in E(P_{T_s}(v_0, v_f))} c(e),$$

where $E(P_{T_s}(v_0, v_f))$ represents the set of edges of the directed path $P_{T_s}(v_0, v_f)$.

In [5,6] is proved that for acyclic c -game on network (G, V_1, V_2, c) there exist the strategies of players s_A^*, s_B^* such that

$$\hat{p}(v) = \hat{F}_v(s_A^*, s_B^*) = \max_{s_A \in S_A} \min_{s_B \in S_B} \hat{F}_v(s_A, s_B) = \min_{s_B \in S_B} \max_{s_A \in S_A} \hat{F}_v(s_A, s_B) \quad (1)$$

and s_A^*, s_B^* do not depend on starting position $v \in V$, i.e. (1) holds for every $v \in V$. Note that here for different positions u and v the values $p(u)$ and $p(v)$ may be different.

The equality (1) is evident in the case when $\text{ext}(c, u) = 0, \forall u \in V$. In this case $p(u) = 0, \forall u \in V$ and the optimal strategies of players can be obtained by fixing the maps $s_A^*: V_A \setminus \{v_f\} \rightarrow V$ and $s_B^*: V_B \setminus \{v_f\} \rightarrow V$ such that $s_A^*(u) \in \text{VEXT}(c, u)$ for $u \in V_A \setminus \{v_f\}$ and $s_B^*(u) \in \text{VEXT}(c, u)$ for $u \in V_B \setminus \{v_f\}$.

If the network (G, V_A, V_B, c) has the property $\text{ext}(c, u) = 0, \forall u \in V \setminus \{v_f\}$ then it is named the network in canonic form [5]. So, for the acyclic c -game on network in canonic form the equality (1) holds and $p(v) = 0, \forall v \in V$.

In general case the equality (1) can be proved using the properties of the potential transformation $c'(u, v) = c(u, v) + \varepsilon(v) - \varepsilon(u)$ on edges $e = (u, v)$ of the network, where $\varepsilon: V \rightarrow R$ is an arbitrary real function on V . The fact is that such transformation of the costs on edges of the network in acyclic c -game do not change the optimal strategies of players, although the values $\hat{p}(v)$ of the positions $v \in V$ are changed by $\hat{p}(v) + \varepsilon(v_f) - \varepsilon(v)$. It means that for arbitrary function $\varepsilon: V \rightarrow R$ the optimal strategies of the players in acyclic c -games on networks (G, V_A, V_B, c) and on (G, V_A, V_B, c') are the same. Using such property in [5] is proved the following theorem.

Theorem 3. *For arbitrary acyclic network (G, V_A, V_B, c) there exists a function $\varepsilon: R \rightarrow R$ which determine a potential transformation $c'(u, v) = c(u, v) + \varepsilon(v) - \varepsilon(u)$ on edges $e = (u, v) \in E$ such that (G, V_A, V_B, c') has the canonic form. The value $\varepsilon(v), v \in V$, which determine $\varepsilon: V \rightarrow R$, can be found by using the following recursive formula*

$$\left\{ \begin{array}{l} \varepsilon(v_f) = 0, \\ \varepsilon(u) = \begin{cases} \max_{v \in V_G(u)} \{c(u, v) + \varepsilon(v)\}, & \text{for } u \in V_A \setminus \{v_f\}, \\ \min_{v \in V_G(u)} \{c(u, v) + \varepsilon(v)\}, & \text{for } u \in V_B \setminus \{v_f\}. \end{cases} \end{array} \right. \quad (2)$$

According to this theorem the optimal strategies of players in acyclic c -game can be found using the following algorithm.

Algorithm 1.

1. Find the values $\varepsilon(v)$, $v \in V$, according to recursive formula (2) and the corresponding potential transformation $c'(u, v) = c(u, v) + \varepsilon(v) - \varepsilon(u)$ for edges $(u, v) \in E$.
2. Fix the arbitrary maps $s_A^*: V_A \setminus \{v_f\} \rightarrow V$ and $s_B^*: V_B \setminus \{v_f\} \rightarrow V$ for which $s_A^*(u) \in VEXT(c', u)$ for $u \in V_A \setminus \{v_f\}$ and $s_B^*(u) \in VEXT(c', u)$ for $u \in V_B \setminus \{v_f\}$.

Remark. The values $\varepsilon(u)$, $u \in V$, represent the values of the acyclic c -game on (G, V_A, V_B, c) with starting position u , i.e. $\varepsilon(u) = \hat{p}(u)$, $\forall u \in V$. Algorithm 1 need $O(n^2)$ elementary operations because such number of operations need the tabulation of values $\varepsilon(u)$, $u \in V$, by using formula (2).

4.2 Acyclic c -Game on Arbitrary Networks

Let us consider the game from section 4.2 when G is an arbitrary graph, i.e. G may contain directed cycles. In this case the subgraph $T_s = (V, F_s)$, generated by given strategies of players

$$s_A: u \rightarrow v \in V_G(u) \text{ for } u \in V_A \setminus \{v_f\}; \quad s_B: u \rightarrow v \in V_G(u) \text{ for } u \in V_B \setminus \{v_f\},$$

may contain directed cycles. So, for given s_A and s_B either exists a unique directed path $P_{T_s}(v_0, v_f)$ from v_0 to v_f in T_s or such a path does not exist in T_s . In the second case there exists a unique directed cycle C_s which can be reached from v_0 through the edges of T_s . If in T_s there exist a unique directed path $P_{T_s}(v_0, v_f)$ from v_0 to v_f then $\hat{F}_{v_0}(s_A, s_B)$ we define as

$$\hat{F}_{v_0}(s_A, s_B) = \sum_{e \in E(P_{T_s}(v_0, v_f))} c(e).$$

If in T_s there are no directed paths from v_0 to v_f then $\hat{F}_{v_0}(s_A, s_B)$ we define in the following way.

Let C_s be the cycle which can be reached from v_0 and $P'_{T_s}(v_0, u_0)$ represent a unique directed path in T_s which connects v_0 with the cycle C_s ($P'_{T_s}(v_0, u_0)$ and C_s have no common edges). Then we put

$$\hat{F}_{v_0}(s_A, s_B) = \begin{cases} +\infty & \text{if } \sum_{e \in E(C_s)} c(e) > 0, \\ \sum_{e \in E(P'_{T_s}(v_0, u_0))} c(e) & \text{if } \sum_{e \in E(C_s)} c(e) = 0, \\ -\infty & \text{if } \sum_{e \in E(C_s)} c(e) < 0. \end{cases}$$

We consider the problem of finding the strategies s_A^*, s_B^* for which

$$\hat{F}_{v_0}(s_A^*, s_B^*) = \max_{s_A \in S_A} \min_{s_B \in S_B} \hat{F}_{v_0}(s_A, s_B).$$

This problem [7] is named the max-min path problem in c -game.

It is easy to observe that for this problem may be the case when the players couldn't reach the final position v_f . We shall formulate the condition when for the player in this game there exists a settle value $\hat{p}(v)$ for each $v \in V$.

Theorem 4. *Let (G, V_A, V_B, c) be a network with given final position, where G is on arbitrary directed graph. Moreover let us consider that $\sum_{e \in E(C_s)} c(e) \neq 0$ for every cycle C_s from G . Then for c -game on (G, V_A, V_B, c) the conditions*

$$\hat{p}(v) = \hat{F}_v(s_A^*, s_B^*) = \max_{s_A \in S_A} \min_{s_B \in S_B} \hat{F}_v(s_A, s_B) = \min_{s_B \in S_B} \max_{s_A \in S_A} \hat{F}_v(s_A, s_B), \forall v \in V$$

holds if and only if there exists a function $\varepsilon: V \rightarrow R$ which determine a potential transformation $c'(u, v) = c(u, v) + \varepsilon(v) - \varepsilon(u)$ on edges $(u, v) \in E$ such that $\text{ext}(c', u) = 0, \forall v \in V$. The optimal strategies of players in c -game on (G, V_A, V_B, c) can be found by fixing the arbitrary maps $s_A^*: V_A \setminus \{v_f\} \rightarrow V$ and $s_B^*: V_B \setminus \{v_f\} \rightarrow V$ such that $s_A^*(u) \in \text{VEXT}(c', u)$ for $u \in V_A$ and $s_B^*(v) \in \text{VEXT}(c', v)$ for $v \in V_B$.

Proof. \Rightarrow Let us consider that the condition

$$\hat{p}(v) = \hat{F}_v(s_A^*, s_B^*) = \max_{s_A \in S_A} \min_{s_B \in S_B} \hat{F}_v(s_A, s_B) = \min_{s_B \in S_B} \max_{s_A \in S_A} \hat{F}_v(s_A, s_B)$$

holds for every $v \in V$ and $\hat{p}(v) < \infty, \forall v \in V$. It is easy to observe that if we put $\varepsilon(v) = \hat{p}(v)$ for $v \in V$ then the function $\varepsilon: V \rightarrow R$ determine a potential transformation $c'(u, v) = c(u, v) + \varepsilon(v) - \varepsilon(u)$ on edges $(u, v) \in E$ such that holds $\text{ext}(c', u) = 0, \forall u \in V$.

\Leftarrow Let us consider that there exists a potential transformation $c'(u, v) = c(u, v) + \varepsilon(v) - \varepsilon(u)$ of costs of edges $(u, v) \in E$ such that $\text{ext}(c', u) = 0, \forall u \in V$. Then for a c' -game on network (G, V_A, V_B, c') the following condition

$$\max_{s_A \in S_A} \min_{s_B \in S_B} \hat{F}'_v(s_A, s_B) = \min_{s_B \in S_B} \max_{s_A \in S_A} \hat{F}'_v(s_A, s_B), \forall v \in V$$

holds. Since the potential transformation do not change the optimal strategies of players then

$$\max_{s_A \in S_A} \min_{s_B \in S_B} \hat{F}_v(s_A, s_B) = \min_{s_B \in S_B} \max_{s_A \in S_A} \hat{F}_v(s_A, s_B), \forall v \in V.$$

Corollary 5. *The difference $\varepsilon(v) - \varepsilon(v_f), v \in V$, in Theorem 4 determine the cost of max-min path from v to v_f , i.e. $\varepsilon(v) = \hat{p}(v), \forall v \in V$.*

Let us consider the network (G, V_A, V_B, c) for which there exist min-max paths from every $v \in V$ to given $v_0 \in V$. Then the following algorithm finds all values $\varepsilon(v)$, $v \in V$, of min-max paths from v to v_0 .

Algorithm 2.

1. We construct an auxiliary acyclic graph $H = (W, \overline{E})$, where

$$W = \{w_0^0\} \cup W^1 \cup W^2 \cup \dots \cup W^n, \quad W^i \cap W^j = \emptyset, i \neq j;$$

$$W^i = \{w_0^i, w_1^i, \dots, w_{n-1}^i\}, \quad i = \overline{1, n};$$

$$\overline{E} = E^0 \cup E^1 \cup E^2 \cup \dots \cup E^{n-1};$$

$$E^i = \{(w_k^{i+1}, w_l^i) \mid (v_k, v_l) \in E\}, \quad i = \overline{1, n-1};$$

$$E^0 = \{(w_k^i, w_0^0) \mid (v_k, v_0) \in E, \quad i = \overline{1, n}\}.$$

The vertex set W of H is obtained from V if it is doubled n times and then a sink vertex w_0^0 is added. The edge subset $E^i \subseteq \overline{E}$ in H connect the vertices of the set W^{i+1} and the vertices of the set W^i in the following way: if in G there exists an edge $(v_k, v_l) \in E$ then in H we add the edge (w_k^{i+1}, w_l^i) . The edge subset $E^0 \subseteq \overline{E}$ in H connect the vertices $w_k^i \in W^1 \cup W^2 \cup \dots \cup W^n$ with the sink vertex w_0^0 , i.e. if there exists an edge $(v_k, v_0) \in E$ then in H we add the edges $(w_k^i, w_0^0) \in E^0$, $i = \overline{1, n}$.

After that we define the acyclic network (H_0, W_A, W_B, c_0) , $H_0 = (W_0, E_0)$ where H_0 is obtained from H by deleting the vertices $w_k^i \in W$ from which the vertex w_0^0 couldn't be attainable. The sets W_A, W_B and the cost function $c_0: E_0 \rightarrow R$ are defined as follows:

$$W_A = \{w_k^i \in W_0 \mid v_k \in V_A\}, \quad W_B = \{w_k^i \in W_0 \mid v_k \in V_B\};$$

$$c_0(w_k^{i+1}, w_k^i) = c(v_k, v_l) \quad \text{if } (v_k, v_l) \in E \text{ and } (w_k^{i+1}, w_k^i) \in E_0;$$

$$c_0(w_k^i, w_0^0) = c(v_k, v_0) \quad \text{if } (v_k, v_0) \in E \text{ and } (w_k^i, w_0^0) \in E_0.$$

2. On auxiliary acyclic network (H_0, W_A, W_B, c_0) with sink vertex w_0^0 we consider the acyclic c -game. According to recursive formula (2) we find $\varepsilon(w_k^i)$ for every $w_k^i \in H_0$, i.e. $\varepsilon(w_k^i) = \hat{p}(w_k^i)$, $\forall w_k^i \in H_0$.

3. Fix $U = \{v_0\}$; $i = 0$.

4. $i = i + 1$.

5. Find the edge set $E(U) = \{(v_k, v_l) \in E \mid v_l \in U, v_k \in V \setminus U\}$ and the vertex set $V(U) = \{v_k \in V \mid e = (v_k, v_l) \in E(U)\}$.

6. Fix $\varepsilon(v_k) = \varepsilon(w_k^i)$ for $v_k \in V(U)$.

7. Select the vertex set $V^0(U)$ from $V(U)$ where $v_k \in V^0(U)$ if the following conditions are satisfied:

a) $c(v_k, v_q) + \varepsilon(v_q) - \varepsilon(v_k) \leq 0$, $\forall v_q \in U \cup V(U), (v_k, v_q) \in E$ if $v_k \in V_A \cap V(U)$;

b) $c(v_k, v_q) + \varepsilon(v_q) - \varepsilon(v_k) \geq 0$, $\forall v_q \in U \cup V(U), (v_k, v_q) \in E$ if $v_k \in V_B \cap V(U)$;

c) there exist an edge $(v_k, v_l) \in E$ such that $c(v_k, v_l) + \varepsilon(v_l) - \varepsilon(v_k) = p(v_0)$.

8. If $V^0(U) = \emptyset$ go to step 10.
9. Change U by $U \cup V^0(U)$.
10. If $U \neq V$ then go to step 4, otherwise go to next step 11.
11. End.

The correctness of this algorithm follow from the algorithm from [8].

4.3 Acyclic l -Game on Networks

Let (G, V_A, V_B, c) be the network with the properties described in section 1. So, $G = (V, E)$ represent a directed graph without directed cycles and G contains a sink vertex $v_f \in V$. On E is defined a function $c: E \rightarrow R$ and on V is given a partition $V = V_A \cup V_B$ ($V_A \cap V_B = \emptyset$) where V_A and V_B are considered as a positions sets of two player.

We study the following acyclic l -game from [5]. Again the strategies of players we define as a maps

$$s_A: u \rightarrow v \in V_G(u) \text{ for } u \in V_A \setminus \{v_f\}; \quad s_B: u \rightarrow v \in V_G(u) \text{ for } u \in V_B \setminus \{v_f\}.$$

The payoff function $\bar{F}_{v_0}: S_A \times S_B \rightarrow R$ in this game we define as follows.

Let $s_A \in S_A$ and $s_B \in S_B$ be a fixed strategies of players. Then the tree $T_s = (V, E_s)$ contains a unique directed path $P_{T_s}(v_0, v_f)$ from v_0 to v_f . We put

$$\bar{F}_{v_0}(s_A, s_B) = \frac{1}{n(P_{T_s}(v_0, v_f))} \sum_{e \in E(P_{T_s}(v_0, v_f))} c(e),$$

where $E(P_{T_s}(v_0, v_f))$ is the set of edges of the path $P_{T_s}(v_0, v_f)$ and $n(P_{T_s}(v_0, v_f))$ is the number of its edges.

In [5,6] is proved that the function $\bar{F}_{v_0}(s_A, s_B)$ satisfy the following condition:

$$\begin{aligned} \bar{p}(v_0) &= \bar{F}_{v_0}(s_A^*, s_B^*) = \max_{s_A \in S_A} \min_{s_B \in S_B} \bar{F}_{v_0}(s_A, s_B) = \\ &= \min_{s_B \in S_B} \max_{s_A \in S_A} \bar{F}_{v_0}(s_A, s_B). \end{aligned} \tag{3}$$

In this game the optimal strategies s_A^* and s_B^* of players depend on starting position v_0 . But the players can achieve the settle value $\bar{p}(v_0)$ starting from v_0 by using the successive choosing of optimal strategies of moves from one position to another while the final position v_f is not reached.

The equality (3) can be proved by using the Theorem 1 (corollary 2). It is easy to observe that if for given acyclic l -game we identify the positions v_0 and v_f we obtain the cyclic game with an auxiliary ergodic network (G', V'_A, V'_B, c') and given starting position v_0 . In this ergodic network the graph G' is obtained from G when the vertices v_0 and v_f are identified. In the graph G' all directed cycles pass through the vertex v_0 . Applying the corollary 1 of Theorem 1 to the obtained cyclic games on ergodic network (G', V'_A, V'_B, c') with starting position v_0 , we obtain the proof of the following theorem [5].

Theorem 6. Let be given an acyclic l -game determined by the network (G, V_A, V_B, c) with starting position v_0 . Then there exist a value $\bar{p}(v_0)$ and a function $\varepsilon: V \rightarrow R$ which determine a potential transformation $c'(u, v) = c(u, v) + \varepsilon(v) - \varepsilon(u)$ of costs on edges $e = (u, v) \in E$ such that the following conditions holds

- a) $\bar{p}(v_0) = \text{ext}(c', u), \forall u \in V \setminus \{v_f\};$
- b) $\varepsilon(v_0) = \varepsilon(v_f)$

the optimal strategies of players in acyclic l -game can be found as follows: fix the arbitrary maps $s_A^*: V_A \setminus \{v_f\} \rightarrow V$ and $s_B^*: V_B \setminus \{v_f\} \rightarrow V$ such that $s_A^*(u) \in \text{VEXT}(c', u)$ for $u \in V \setminus \{v_f\}$ and $s_B^*(u) \in \text{VEXT}(c', u)$.

This theorem follow from corollary 2 of Theorem 1 because the acyclic l -game on network Let (G, V_A, V_B, c) with starting position v_0 can be regard as a cyclic game on network Let (G', V'_A, V'_B, c') with starting position v_0 . This network is obtained from the network (G, V_A, V_B, c) when the vertices v_0 and v_f are identified.

Taking in consideration remark to Algorithm 1 and corollary 5 of Theorem 4 we obtain the following result.

Corollary 7. The difference $\varepsilon(v) - \varepsilon(v_0), v \in V$, in Theorem 6 represent the costs of max-min path from v to v_f in acyclic c -game on network (G, V_A, V_B, \bar{c}) where $\bar{c}(u, v) = c(u, v) - \bar{p}(v_0), \forall (u, v) \in E$.

So, if $\bar{p}(v_0)$ is known then the optimal strategies s_A^* and s_B^* of players in acyclic l -games can be found in time $O(n^2)$ by reducing acyclic l -game to acyclic c -game on (G, V_A, V_B, \bar{c}) .

On the basis of this property in [5] is elaborated polynomial algorithm for solving l -games. In [6] is made an attempt to elaborate strongly polynomial time algorithm for l -games. Unfortunately the algorithm from [6] is not valid.

5 Polynomial Time Algorithm for Solving Cyclic Games

On the basis of the obtained results we can propose polynomial time algorithm for solving cyclic games.

We consider an acyclic game on ergodic network (G, V_A, V_B, c) with given starting position v_0 . The graph $G = (V, E)$ is considered to be strongly connected and $V = \{v_0, v_1, v_2, \dots, v_{n-1}\}$. Assume that v_0 belong to the cycle C_{s^*} determined by the optimal strategies of players s_A^* and s_B^* .

We construct an auxiliary acyclic graph $H_r = (\overline{W}_r, \overline{E}_r)$, where

$$\overline{W}_r = \{w_0^0\} \cup W^1 \cup W^2 \cup \dots \cup W^r, \quad W^i \cap W^j = \emptyset, \quad i \neq j;$$

$$W^i = \{w_0^i, w_1^i, \dots, w_{n-1}^i\}, \quad i = \overline{1, r};$$

$$\overline{E} = E^0 \cup E^1 \cup E^2 \cup \dots \cup E^{r-1};$$

$$E^i = \{(w_k^{i+1}, w_l^i) | (v_k, v_l) \in E\}, \quad i = \overline{1, r-1};$$

$$E^0 = \{(w_k^i, w_0^0) | (v_k, v_0) \in E, \quad i = \overline{1, r}\}.$$

The vertex set W_r of H_r is obtained from V if it is doubled r times and then a sink vertex w_0^0 is added. The edge subset $E^i \subseteq \overline{E}$ in H_r connect the vertices of the set W^{i+1} and the vertices of the set W^i in the following way: if in G there exists an edge $(v_k, v_l) \in E$ then in H_r we add the edge (w_k^{i+1}, w_l^i) . The edge subset $E^0 \subseteq \overline{E}$ in H_r connect the vertices $w_k^i \in W^1 \cup W^2 \cup \dots \cup W^r$ with the sink vertex w_0^0 , i.e. if there exists an edge $(v_k, v_0) \in E$ then in H_r we add the edges $(w_k^i, w_0^0) \in E^0$, $i = \overline{1, r}$.

After that we define the acyclic network (H'_r, W_A, W_B, c'_r) , $H'_r = (W_r, E_r)$ where H'_r is obtained from H_r by deleting the vertices $w_k^i \in \overline{W}_r$ from which the vertex w_0^0 couldn't be attainable. The sets W_A, W_B and the cost function $c'_r: E_r \rightarrow R$ are defined as follows:

$$\begin{aligned} W_A &= \{w_k^i \in W_0 \mid v_k \in V_A\}, \quad W_B = \{w_k^i \in W_0 \mid v_k \in V_B\}; \\ c'_r(w_k^{i+1}, w_k^i) &= c(v_k, v_l) \quad \text{if } (v_k, v_l) \in E \quad \text{and} \quad (w_k^{i+1}, w_k^i) \in E_r; \\ c'_r(w_k^i, w_0^0) &= c(v_k, v_0) \quad \text{if } (v_k, v_0) \in E \quad \text{and} \quad (w_k^i, w_0^0) \in E_r. \end{aligned}$$

Now we consider the acyclic c -game on acyclic network (H'_r, W_A, W_B, c'_r) with sink vertex w_0^0 and starting position w_0^r .

Lemma 8. *Let $p = p(v_0)$ is a value of the ergodic cyclic game on G and the number of edges of the max-min cycle in G is equal r . Moreover, let $p_r(W_0^r)$ be the value of the l -game on (H'_r, W_A, W_B, c'_r) with starting position w_0^r . Then $p(v_0) = p_r(w_0^r)$.*

Proof. It is evident that there exist a bijective mapping between the set of cycles with exactly r edges (which contains the vertex v_0) in G and the set of directed paths with r edges from w_0^r to w_0^0 in H'_r . Therefore $p(v_0) = p_r(w_0^r)$.

On the basis of this lemma we can propose the following algorithm for finding the optimal strategies of players in cyclic games.

Algorithm 3.

We construct the acyclic networks (H'_r, W_A, W_B, c'_r) , $r = 2, 3, \dots, n$ and for each of them solve l -game. In a such way we find the values $p_2(w_0^2), p_3(w_0^3), \dots, p_n(w_0^n)$ for these l -games. Then we consequetively fix $p = p_2(w_0^2), p_3(w_0^3), \dots, p_n(w_0^n)$ and each time solve the c -game on network (G, V_A, V_B, c') , where $c' = c - p$. Fixing each time the values $\varepsilon'(v_k) = \tilde{p}'(v_k)$ for $v_k \in V$ we check if the following condition

$$\text{ext}(\overline{c}', v_k) = 0. \quad \forall v_k \in V$$

is satisfied, where $\overline{c}'(v_k, v_l) = c'(v_k, v_l) + \varepsilon(v_l) - \varepsilon(v_k)$. We find r for which this condition holds and fix the respective maps s_A^* and s_B^* such that $s_A^*(v_k) \in \text{VEXT}(c', v_k)$ for $u \in V_A$ and $s_B^*(v_k) \in \text{VEXT}(c', v_k)$ for $u \in V_B$. So, s_A^* and s_B^* represent the optimal strategies of player in cyclic games on G .

Remark. Algorithm 3 find the value $p(v_0)$ and optimal strategies of players in time $O(n^2|E|^2)$.

References

1. A. Ehrenfeucht, J. Mycielski, Positional strategies for mean payoff games. *International Journal of Game Theory*, (8), 1979, p. 109–113.
2. V.A. Gurvich, A.V. Karzanov, L.G. Khachiyan, Cyclic games and an algorithm to find minmax cycle means in directed graphs. USSR, *Computational Mathematics and Mathematical Physics*, (28), 1988, p. 85–91.
3. D.D. Lozovanu, Extremal-Combinatorial problems and algorithms for its solving. Kishinev, Stiinta, 1991.
4. U. Zwick, M. Paterson, The complexity of mean payoff games on graphs. *TCS*, 158: 344–359, 1996.
5. D.D. Lozovanu, Algorithms to solve some classes of network minmax problems and their applications, *Cybernetics* (29), 1991, p. 93–100.
6. D.D. Lozovanu, Strongly polynomial algorithms for finding minimax paths in networks and solution of cyclic games, *Cybernetics and Systems Analysis*, (29), 1993, p. 754–759.
7. D.D. Lozovanu, V.A. Trubin, Min-max path problem on network and an algorithm for its solving, *Discrete Mathematics and Applications*, (6), 1994, p. 138–144.
8. R. Boliac, D. Lozovanu, D. Solomon, Optimal paths in network games with p players, *Discrete Applied Mathematics*, vol. 99, N 1–3 (2000), p. 339–348.

A Robust Optimization Approach to Supply Chain Management

Dimitris Bertsimas and Aurélie Thiele

Massachusetts Institute of Technology, Cambridge MA 02139,
`dbertsim@mit.edu, aurelie@mit.edu`

Abstract. We propose a general methodology based on robust optimization to address the problem of optimally controlling a supply chain subject to stochastic demand in discrete time. The attractive features of the proposed approach are: (a) It incorporates a wide variety of phenomena, including demands that are not identically distributed over time and capacity on the echelons and links; (b) it uses very little information on the demand distributions; (c) it leads to qualitatively similar optimal policies (basestock policies) as in dynamic programming; (d) it is numerically tractable for large scale supply chain problems even in networks, where dynamic programming methods face serious dimensionality problems; (e) in preliminary computational experiments, it often outperforms dynamic programming based solutions for a wide range of parameters.

1 Introduction

Optimal supply chain management has been extensively studied in the past using dynamic programming, which leads to insightful policies for simpler systems (basestock policies for series systems; Clark and Scarf [7]). Unfortunately, dynamic programming assumes complete knowledge of the probability distributions and suffers from the curse of dimensionality. As a result, preference for implementation purposes is given to more intuitive policies that are much easier to compute, but also suboptimal (see Zipkin [12]).

Hence, the need arises to develop a new optimization approach that incorporates the stochastic character of the demand in the supply chain without making any assumptions on its distribution, is applicable to a wide range of network topologies, is easy to understand intuitively, and combines computational tractability with the structural properties of the optimal policy. The goal of this paper is to present such an approach, based on robust linear and mixed integer optimization that has witnessed increased research activity (Soyster [11], Ben-Tal and Nemirovski ([1,2,3]) and El-Ghaoui et. al. ([8,9], Bertsimas and Sim [5,6])). We utilize the approach in [5,6], which leads to linear robust counterparts while controlling the level of conservativeness of the solution.

The contributions of this paper are as follows: (a) We develop an approach that incorporates demand uncertainty in a deterministic manner, remains numerically tractable as the dimension of the problem increases and leads to high-quality solutions without assuming a specific demand distribution. (b) The robust problem is of the same class as the nominal problem, that is, a linear

programming problem if there are no fixed costs or a mixed integer programming problem if fixed costs are present, independently of the topology of the network. (c) The optimal robust policy is qualitatively similar to the optimal policy obtained by dynamic programming when known. In particular, it remains basestock when the optimal stochastic policy is basestock, as well as in some other cases where the optimal stochastic policy is not known. (d) We derive closed-form expressions of key parameters defining the optimal policy. These expressions provide a deeper insight into the way uncertainty affects the optimal policy in supply chain problems.

2 The Robust Optimization Approach

We rely extensively on the robust optimization tools developed by Bertsimas and Sim in [5] for linear programming problems. We consider the following problem subject to data uncertainty:

$$\min \mathbf{c}'\mathbf{x} : \mathbf{Ax} \leq \mathbf{b}, \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},$$

where we assume WLOG that only the matrix \mathbf{A} is subject to data uncertainty.

$$\text{Let } \mathcal{A} = \left\{ \mathbf{A} \in \mathcal{R}^{m \times n} \mid a_{ij} \in [\bar{a}_{ij} - \hat{a}_{ij}, \bar{a}_{ij} + \hat{a}_{ij}] \forall i, j, \sum_{(i,j) \in J} \frac{|a_{ij} - \bar{a}_{ij}|}{\hat{a}_{ij}} \leq \Gamma \right\}.$$

Γ is a parameter that controls the degree of conservatism. The robust problem is then formulated as:

$$\begin{aligned} & \min \mathbf{c}'\mathbf{x} \\ & \text{s.t. } \mathbf{Ax} \leq \mathbf{b}, \quad \forall \mathbf{A} \in \mathcal{A} \\ & \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \end{aligned} \tag{1}$$

Theorem 1 (Bertsimas and Sim [5]). *The uncertain linear programming problem has the following robust, linear counterpart:*

$$\begin{aligned} & \min \mathbf{c}'\mathbf{x} \\ & \text{s.t. } \sum_j \bar{a}_{ij}x_j + q_i\Gamma + \sum_{j:(i,j) \in J} r_{ij} \leq b_i, \quad \forall i \\ & \quad q_i + r_{ij} \geq \hat{a}_{ij}y_j, \quad \forall (i, j) \in J \\ & \quad -\mathbf{y} \leq \mathbf{x} \leq \mathbf{y} \\ & \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & \quad \mathbf{q} \geq 0, \quad \mathbf{r} \geq 0, \quad \mathbf{y} \geq 0. \end{aligned} \tag{2}$$

The robust counterpart is therefore of the same class as the nominal problem, that is, a linear programming problem. This is a highly attractive feature of this approach, since linear programming problems are readily solved by standard optimization packages. Moreover, if in the original problem (1), some of the variables were constrained to be integers, then the robust counterpart (2) would retain the same properties, i.e., the robust counterpart of a mixed integer programming problem is itself another mixed integer programming problem.

3 The Single Station Case

3.1 The Uncapacitated Model

In this section we apply the robust optimization framework to the problem of ordering, at a single installation, a single type of item subject to stochastic demand over a finite discrete horizon of T periods, so as to minimize a given cost function. We define, for $k = 0, \dots, T$:

x_k : the stock available at the beginning of the k th period,

u_k : the stock ordered at the beginning of the k th period,

w_k : the demand during the k th period.

The stock ordered at the beginning of the k th period is delivered before the beginning of the $(k+1)$ st period, that is, all orders have a constant leadtime equal to 0. Excess demand is backlogged. Therefore, the evolution of the stock over time is described by the following linear equation:

$$x_{k+1} = x_k + u_k - w_k, \quad k = 0, \dots, T-1, \quad (3)$$

leading to the closed-form expression:

$$x_{k+1} = x_0 + \sum_{i=0}^k (u_i - w_i), \quad k = 0, \dots, T-1. \quad (4)$$

Neither the stock available nor the quantity ordered at each period are subject to upper bounds. Section 3.2 deals with the capacitated case.

The demands w_k are random variables. In order to apply the approach outlined in Sect. 2, we model w_k for each k as an uncertain parameter that takes values in $[\bar{w}_k - \hat{w}_k, \bar{w}_k + \hat{w}_k]$. We define the scaled deviation of w_k from its nominal value to be $z_k = (w_k - \bar{w}_k)/\hat{w}_k$, which takes values in $[-1, 1]$. We impose budgets of uncertainty at each time period k for the scaled deviations up to time k . Hence, we now have the constraint $\sum_{i=0}^k |z_i| \leq \Gamma_k$ for all time periods $k = 0, \dots, T-1$. These budgets of uncertainty rule out large deviations in the cumulative demand, and as a result the robust methodology can be understood as a “reasonable worst-case” approach. The main assumption we make on the Γ_k is that they are increasing in k , i.e., we feel that uncertainty increases with the number of time periods considered. We also constrain the Γ_k to be increasing by at most 1 at each time period, i.e., the increase of the budgets of uncertainty should not exceed the number of new parameters added at each time period.

Finally, we specify the cost function. The cost incurred at period k consists of two parts: a purchasing cost $C(u_k)$ and a holding/shortage cost resulting from this order $R(x_{k+1})$. Here, we consider a purchasing cost of the form:

$$C(u) = \begin{cases} K + c \cdot u, & \text{if } u > 0, \\ 0, & \text{if } u = 0, \end{cases} \quad (5)$$

with $c > 0$ the unit variable cost and $K \geq 0$ the fixed cost. If $K > 0$, a fixed positive cost is incurred whenever an order is made. The holding/shortage

cost represents the cost associated with having either excess inventory (positive stock) or unfilled demand (negative stock). We consider a convex, piecewise linear holding/shortage cost:

$$R(x) = \max(hx, -px), \quad (6)$$

where h and p are nonnegative. The holding/shortage cost for period k , y_k , is computed at the end of the period, after the shipment u_k has been received and the demand w_k has been realized. We assume $p > c$, so that ordering stock remains a possibility up to the last period.

Using the piecewise linearity and convexity of the holding/shortage cost function, and modelling the fixed ordering cost with binary variables, the inventory problem we consider can be written as a mixed integer programming problem:

$$\begin{aligned} & \min \sum_{k=0}^{T-1} (cu_k + Kv_k + y_k) \\ \text{s.t. } & y_k \geq h \left(x_0 + \sum_{i=0}^k (u_i - w_i) \right) \quad k = 0, \dots, T-1 \\ & y_k \geq -p \left(x_0 + \sum_{i=0}^k (u_i - w_i) \right) \quad k = 0, \dots, T-1 \\ & 0 \leq u_k \leq Mv_k, \quad v_k \in \{0, 1\} \quad k = 0, \dots, T-1, \end{aligned} \quad (7)$$

where $w_i = \bar{w}_i + \hat{w}_i \cdot z_i$ such that $\mathbf{z} \in \mathcal{P} = \{|z_i| \leq 1 \ \forall i \geq 0, \ \sum_{i=0}^k |z_i| \leq \Gamma_k \ \forall k \geq 0\}$. Applying Theorem 1, we obtain:

Theorem 2. *The robust formulation for the single-station inventory problem (7) is:*

$$\begin{aligned} & \min \sum_{k=0}^{T-1} (cu_k + Kv_k + y_k) \\ \text{s.t. } & y_k \geq h \left(x_0 + \sum_{i=0}^k (u_i - \bar{w}_i) + q_k \Gamma_k + \sum_{i=0}^k r_{ik} \right) \\ & y_k \geq p \left(-x_0 - \sum_{i=0}^k (u_i - \bar{w}_i) + q_k \Gamma_k + \sum_{i=0}^k r_{ik} \right) \\ & q_k + r_{ik} \geq \hat{w}_i \\ & q_k \geq 0, \quad r_{ik} \geq 0 \\ & 0 \leq u_k \leq Mv_k, \quad v_k \in \{0, 1\}, \end{aligned} \quad (8)$$

where M is a large positive number.

The variables q_k and r_{ik} quantify the sensitivity of the cost to infinitesimal changes in the key parameters of the robust approach, specifically the level of conservativeness and the bounds of the uncertain variables. $q_k \Gamma_k + \sum_{i=0}^k r_{ik}$ represents the extra inventory (or lack thereof) that we want to take into account in controlling the system from a worst-case perspective.

The robust problem is a linear programming problem if there is no fixed cost ($K = 0$) and a mixed integer programming problem if fixed costs are present ($K > 0$). In both cases, this robust model can readily be solved numerically through standard optimization tools, which is of course very appealing. It is also desirable to have some theoretical understanding of the optimal policy, in particular with respect to the optimal nominal policy and, if known, the optimal stochastic policy. We address these questions next.

Definition 1 ((S,S) and (s,S) policies). *The optimal policy of a discrete-horizon inventory problem is said to be (s, S) , or basestock, if there exists a threshold sequence (s_k, S_k) such that, at each time period k , it is optimal to order $S_k - x_k$ if $x_k < s_k$ and 0 otherwise, with $s_k \leq S_k$. If there is no fixed ordering cost ($K = 0$), $s_k = S_k$.*

In order to analyze the optimal robust policy, we need the following lemma:

Lemma 1. (a) *The optimal policy in the stochastic case, where the cost to minimize is the expected value of the cost function over the random variables w_k , is (s, S) . As a result, the optimal policy for the nominal problem is also (s, S) .*

(b) *For the nominal problem without fixed cost, the optimal policy for the nominal case is (S, S) with the threshold at time k being $S_k = \bar{w}_k$.*

(c) *For the nominal problem with fixed cost, if we denote by t_j ($j = 1, \dots, J$) the times where stock is ordered and s_j, S_j the corresponding thresholds at time t_j , we have:*

$$S_j = \sum_{i=0}^{I_j} \bar{w}_{t_j+i}, \quad (9)$$

and

$$s_1 = x_0 - \sum_{i=0}^{t_1-1} \bar{w}_i, \quad s_j = - \sum_{i=I_{j-1}+1}^{L_{j-1}-1} \bar{w}_{t_{j-1}+i}, \quad j \geq 2, \quad (10)$$

where $L_j = t_{j+1} - t_j$ and $I_j = \left\lfloor \frac{pL_j - c \mathbf{1}_{\{j=J\}}}{h+p} \right\rfloor$.

We next present the main result regarding the structure of the optimal robust policy.

Theorem 3 (Optimal robust policy).

(a) *The optimal policy in the robust formulation (8), evaluated at time 0 for the rest of the horizon, is the optimal policy for the nominal problem with the modified demand:*

$$w'_k = \bar{w}_k + \frac{p-h}{p+h} (A_k - A_{k-1}), \quad (11)$$

where $A_k = q_k^* \Gamma_k + \sum_{i=0}^k r_{ik}^*$ is the deviation of the cumulative demand from its mean at time k , \mathbf{q}^* and \mathbf{r}^* being the optimal \mathbf{q} and \mathbf{r} variables in (8). (By convention $q_{-1} = r_{-, -1} = 0$.) In particular it is (S, S) if there is no fixed cost and (s, S) if there is a fixed cost.

- (b) If there is no fixed cost, the optimal robust policy is (S, S) with $S_k = w'_k$ for all k .
- (c) If there is a fixed cost, the corresponding thresholds S_j, s_j , where $j = 1, \dots, J$ indexes the ordering times, are given by (9) and (10) applied to the modified demand w'_k .
- (d) The optimal cost of the robust problem (8) is equal to the optimal cost for the nominal problem with the modified demand plus a term representing the extra cost incurred by the robust policy, $\frac{2ph}{p+h} \sum_{k=0}^{T-1} A_k$.

Proof. Let $(\mathbf{u}^*, \mathbf{v}^*, \mathbf{q}^*, \mathbf{r}^*)$ be the optimal solution of (8). Obviously, setting the \mathbf{q} and \mathbf{r} variables to their optimal values \mathbf{q}^* and \mathbf{r}^* in (8) and resolving the linear programming problem will give \mathbf{u}^* and \mathbf{v}^* again. This enables us to focus on the optimal ordering policy only, taking the auxiliary variables $\mathbf{q}^*, \mathbf{r}^*$ as given in the robust formulation (8). We have then to solve:

$$\min_{\mathbf{u} \geq 0} \sum_{k=0}^{T-1} [cu_k + K1_{\{u_k > 0\}} + \max(h(\bar{x}_{k+1} + A_k), p(-\bar{x}_{k+1} + A_k))] \quad (12)$$

where $\bar{x}_{k+1} = x_0 + \sum_{i=0}^k (u_i - \bar{w}_i)$ and $A_k = q_k^* \Gamma_k + \sum_{i=0}^k r_{ik}^*$ for all k .

We define a modified stock variable x'_k , which evolves according to the linear equation:

$$x'_{k+1} = x'_k + u_k - \underbrace{\left(\bar{w}_k + \frac{p-h}{p+h} (A_k - A_{k-1}) \right)}_{=w'_k}, \quad (13)$$

with $x'_0 = x_0$. Note that the modified demand w'_k is not subject to uncertainty. We have:

$$\max(h(\bar{x}_{k+1} + A_k), p(-\bar{x}_{k+1} + A_k)) = \max(hx'_{k+1}, -px'_{k+1}) + \frac{2ph}{p+h} A_k. \quad (14)$$

The reformulation of the robust model, given the optimal \mathbf{q}^* and \mathbf{r}^* variables, as a nominal inventory problem in the modified stock variable x'_k (plus the fixed cost $\frac{2ph}{p+h} \sum_{k=0}^{T-1} A_k$) follows from injecting (14) into (12). This proves (a) and (d). We conclude that (b) and (c) hold by invoking Lemma 1. \square

Remark: For the case without fixed cost, and for the case with fixed cost when the optimal ordering times are given, the robust approach leads to the thresholds in closed form. For instance, if the demand is i.i.d. ($\bar{w}_k = \bar{w}$, $\hat{w}_k = \hat{w}$ for all k), we have $A_k = \hat{w} \Gamma_k$ and, if there is no fixed cost, $S_k = w'_k = \bar{w} + \frac{p-h}{p+h} \hat{w} (\Gamma_k - \Gamma_{k-1})$ for all k .

Hence, the robust approach protects against the uncertainty of the demand while maintaining striking similarities with the nominal problem, remains computationally tractable and is easy to understand intuitively.

3.2 The Capacitated Model

So far, we have assumed that there was no upper bound either on the amount of stock that can be ordered or on the amount of stock that can be held in the facility. In this section, we consider the more realistic case where such bounds exist. The other assumptions remain the same as in Sect. 3.1.

The Model with Capacitated Orders. The extension of the robust model to capacitated orders of maximal size d is immediate, by adding the constraint:

$$u_k \leq d, \quad \forall k, \quad (15)$$

to (8). We next study the structure of the optimal policy.

Theorem 4 (Optimal robust policy). *The optimal robust policy is the optimal policy for the nominal problem with capacity d on the links and with the modified demand defined in (11). As a result, the optimal policy remains (S, S) (resp (s, S)) in the case without (resp with) fixed cost.*

The Model with Capacitated Inventory. We now consider the case where stock can only be stored up to an amount C . This adds the following constraint to (8):

$$x_0 + \sum_{i=0}^k (u_i - w_i) \leq C, \quad (16)$$

where $w_i = \bar{w}_i + \hat{w}_i \cdot z_i$ such that $\mathbf{z} \in \{|z_i| \leq 1 \quad \forall i, \quad \sum_{i=0}^k |z_i| \leq \Gamma_k \quad \forall k\}$. This constraint depends on the uncertain parameters w_i . Applying the technique developed in Sect. 2, we rewrite this constraint in the robust framework as:

$$\bar{x}_{k+1} + q_k \Gamma_k + \sum_{i=0}^k r_{ik} \leq C, \quad \forall k, \quad (17)$$

where q_k and r_{ik} are defined in (8). We next study the optimal policy.

Theorem 5 (Optimal robust policy). *The optimal robust policy is the optimal policy for the nominal problem subject to the modified demand defined in (11), and with inventory capacity at time 0 equal to C , and inventory capacity at time $k+1$, $k \geq 0$, equal to $C - \frac{2p}{p+h} A_k$.*

As a result, the optimal policy remains (S, S) (resp (s, S)) in the case without (resp with) fixed purchasing cost.

4 The Network Case

4.1 The Uncapacitated Model

We now extend the results of Sect. 3 to the network case. We first study the case of tree networks, which are well suited to describe supply chains because of their hierarchical structure: the main storage hubs (the sources of the network) receive their supplies from outside manufacturing plants and send items throughout the network, each time bringing them closer to their final destination, until they reach the stores (the sinks of the network). Let S be the number of sink nodes. When there is only one sink node, the tree network is called a series system.

We define echelon k , for $k = 1, \dots, N$ with N the total number of nodes in the network, to be the union of all the installations, including k itself, that can receive stock from installation k , and the links between them. In the special case of series systems, we number the installations such that for $k = 1, \dots, N$, the items transit from installation $k + 1$ to k , with installation N receiving its supply from the plant and installation 1 being the only sink node, as in [7]. In that case, the demand at installation $k + 1$ at time t is the amount of stock ordered at installation k at the same time t . We also define, for $k = 1, \dots, N$:

$I_k(t)$: the stock available at the beginning of period t at installation k ,

$X_k(t)$: the stock available at the beginning of period t at echelon k ,

$D_{i_k k}(t)$: the stock ordered at the beginning of period t at echelon k to its supplier i_k ,

$W_s(t)$: the demand at sink node s during period t , $s = 1, \dots, S$.

Let $N(k)$ be the set of installations supplied by installation k and $O(k)$ the set of sink nodes in echelon k . We assume constant leadtimes equal to 0, backlog of excess demand, and linear dynamics for the stock at installation k at time $t = 0, \dots, T - 1$:

$$I_k(t + 1) = I_k(t) + D_{i_k k}(t) - \sum_{j \in N(k)} D_{kj}(t), \quad (18)$$

By convention, if k is a sink node s , $\sum_{j \in N(k)} D_{kj}(t) = W_s(t)$. This leads to the following dynamics for the stock at echelon k at time $t = 0, \dots, T - 1$:

$$X_k(t + 1) = X_k(t) + D_{i_k k}(t) - \sum_{s \in O(k)} W_s(t). \quad (19)$$

Furthermore, the stock ordered by echelon k at time t is subject to the coupling constraint:

$$\sum_{i \in N(k)} D_{ki}(t) \leq \max(I_k(t), 0), \quad \forall k, \forall t, \quad (20)$$

that is, the total order made to a supplier cannot exceed what the supplier has currently in stock, or, equivalently, the supplier can only send through the network items that it really has. Since the network was empty when it started operating at time $t_0 = -\infty$, it follows by induction on t that $I_k(t) \geq 0$ for all

$k \geq 2$. Therefore the coupling constraint between echelons is linear and can be rewritten as:

$$\sum_{i \in N(k)} D_{ki}(t) \leq \bar{X}_k(t) - \sum_{i \in N(k)} \bar{X}_i(t), \quad \forall k, \forall t. \quad (21)$$

Finally, we specify the cost function. We assume that each echelon k has the same cost structure as the single installation modelled in Sect. 3.1 with specific parameters (c_k, K_k, h_k, p_k) . We also keep here the same notations and assumptions as in Sect. 3.1 regarding the uncertainty structure at each sink node. In particular, each sink node s has its own threshold sequence $\Gamma_s(t)$ evolving over time that represents the total budget of uncertainty allowed up to time t for sink s . We have $W_s(t) = \bar{W}_s(t) + \widehat{W}_s(t) \cdot Z_s(t)$ such that the $Z_s(t)$ belong to the set $\mathcal{P}_s = \{|Z_s(t)| \leq 1 \forall t, \sum_{\tau=0}^t Z_s(\tau) \leq \Gamma_s(t), \forall t\}$. We assume $0 \leq \Gamma_s(t) - \Gamma_s(t-1) \leq 1$ for all s and t , that is, the budgets of uncertainty are increasing in t at each sink node, but cannot increase by more than 1 at each time period.

Applying the robust approach developed in Sect. 2 to the holding/shortage constraints in the same manner as in Sect. 3, we obtain the mixed integer programming problem:

$$\begin{aligned} & \min \sum_{t=0}^{T-1} \sum_{k=1}^N \sum_{i \in N(k)} \{c_{ki} D_{ki}(t) + K_{ki} V_{ki}(t) + Y_i(t)\} \\ \text{s.t. } & Y_i(t) \geq h_i \left\{ \bar{X}_i(t+1) + \sum_{s \in O(i)} \left(q_s(t) \Gamma_s(t) + \sum_{\tau=0}^t r_s(\tau, t) \right) \right\}, \\ & Y_i(t) \geq p_i \left\{ -\bar{X}_i(t+1) + \sum_{s \in O(i)} \left(q_s(t) \Gamma_s(t) + \sum_{\tau=0}^t r_s(\tau, t) \right) \right\}, \quad (22) \\ & \sum_{i \in N(k)} D_{ki}(t) \leq \bar{X}_k(t) - \sum_{i \in N(k)} \bar{X}_i(t), \\ & q_s(t) + r_s(\tau, t) \geq \widehat{W}_s(\tau), \\ & q_s(t) \geq 0, \quad r_s(\tau, t) \geq 0, \\ & 0 \leq D_{ki}(t) \leq M V_{ki}(t), \quad V_{ki}(t) \in \{0, 1\}, \end{aligned}$$

with $\bar{X}_i(t+1) = X_i(0) + \sum_{\tau=0}^t \left\{ D_{ki}(\tau) - \sum_{s \in O(i)} \bar{W}_s(\tau) \right\}$ for all i and t , where k supplies i .

As in the single-station case, an attractive feature of this approach is that the robust model of a supply chain remains of the same class as the nominal model, that is, a linear programming problem if there are no fixed costs and a mixed integer programming problem if fixed costs are present. Therefore, the proposed methodology is numerically tractable for very general topologies. The main result is as follows.

Theorem 6 (Optimal robust policy).

(a) The optimal policy in (22) for echelon k is the optimal policy obtained for the supply chain subject to the modified, deterministic demand at sink node s (for $s \in O(k)$):

$$W'_{s,k}(t) = \overline{W}_s(t) + \frac{p_k - h_k}{p_k + h_k} (A_s(t) - A_s(t-1)), \quad (23)$$

where $A_s(t) = q_s^*(t)\Gamma_s(t) + \sum_{\tau=0}^t r_s^*(\tau, t)$, \mathbf{q}_s^* and \mathbf{r}_s^* being the optimal \mathbf{q} and \mathbf{r} variables associated with sink node s in (22).

(b) The optimal cost in the robust case for the tree network is equal to the optimal cost of the nominal problem for the modified demands, plus a term representing the extra cost incurred by the robust policy,

$$\sum_{k=1}^N \frac{2p_k h_k}{p_k + h_k} \sum_{t=0}^{T-1} \sum_{s \in O(k)} A_s(t).$$

The case of more general supply chains is complex because they cannot be reduced to a tree network: the need might arise to order from a more expensive supplier when the cheapest one does not have enough inventory. We can still define echelons for those networks in a similar manner as before, and the evolution of the stock at echelon k , which is supplied by the set of installations $I(k)$ and has the set $O(k)$ as its sink nodes, is described by the following linear equation:

$$X_k(t+1) = X_k(0) + \sum_{\tau=0}^t \left\{ \sum_{i \in I(k)} D_{ik}(\tau) - \sum_{j \in O(k)} W_j(\tau) \right\}. \quad (24)$$

With the standard cost assumptions used before, the echelons cannot be studied independently and the optimal policy is not necessarily basestock, even in the simple case of demand without uncertainty. This is illustrated by the following example.

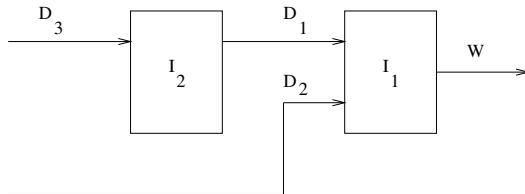


Fig. 1. A network for which the optimal policy is not basestock.

The network in Fig. 1 has two installations, and therefore two echelons. Echelon 1 can be supplied by installation 2 at a unit cost $c_1 = 1$, without any fixed ordering cost, and has the option to order directly from the plant for

the same unit cost $c_2 = 1$, but with an additional fixed cost $K_2 = 4$ incurred whenever an order is made. This option is attractive only if installation 2 does not have enough stock in inventory. The holding and shortage unit costs at echelon 1 are $h_1 = p_1 = 2$. The horizon is 1 time period, and the demand at time 0 is deterministic, equal to 10 units. Echelon 1 has only 5 units in inventory at time 0.

Comparing the two options, it is easy to see that it is optimal for echelon 1 to order 5 units from 2 if 2 has 5 units in inventory at time 0, 2 units from 2 and none from the plant if 2 has 2 units, and 5 units from the plant if 2 has no item in inventory. Therefore, the optimal amount of stock on hand and on order at echelon 1 at time 0 is 10, resp. 7, 10, units if installation 2 has 5, resp 2, 0 units in inventory at time 0. Thus, the optimal policy is not basestock.

Also, while we can reformulate the robust problem as a new problem with modified demand in the same fashion as before, it loses some of its meaning since distinct echelons can now “see” the same sink node but different demands at this node (because of the cost parameters specific to each echelon, which appear in the expression of the modified demand). Hence, it is not clear how they can work together to meet this demand optimally.

However, the proposed robust methodology remains numerically tractable in a wide range of settings, in particular with a holding/shortage cost at the installation level instead of the echelon. This illustrates the applicability of the proposed approach to different cost structures.

4.2 The Capacitated Model

We now refine our description of the inventory problem in a supply chain by introducing upper bounds on the amount of stock that can be ordered and/or held at any time and at any echelon. As explained in Sect. 3.2, an upper bound on the maximal order can be directly introduced in the proposed approach, by adding the constraint:

$$D_{ki}(t) \leq d_{ki} \quad \forall k, \forall i \in \mathcal{N}(k), \forall t, \quad (25)$$

to (22). Inventory capacity, however, requires further manipulation, since the level of inventory held at an echelon at any time depends on the demand, which is subject to uncertainty. Similar manipulations as in Sect. 3.2 lead to the constraint $\forall k, \forall t$:

$$\bar{X}_k(t+1) + \sum_{s \in \mathcal{O}(k)} \left(q_s(t) \Gamma_s(t) + \sum_{\tau=0}^t r_s(\tau, t) \right) \leq C_k \quad (26)$$

to be added to the formulation, $q(t)$ and $r(\tau, t)$ being defined as in (23).

We next study the structure of the optimal policy.

Theorem 7. *The optimal policy at each echelon remains basestock in presence of link and echelon capacities. It is identical to the optimal policy of a nominal problem at a single station subject to the modified demand defined in (23), time-varying echelon capacities: $C'_k(t+1) = C_k - \frac{2p_k}{p_k + h_k} \sum_{s \in \mathcal{O}(k)} A_s(t)$, where C_k is*

the original capacity at echelon k , and link capacities that incorporate $D_{ki}(t) \leq d_{ki}$ for all $k, i \in \mathcal{N}(k)$ and t , as well as the capacity induced by the coupling constraint (21).

5 Numerical Implementation

We now apply the proposed methodology to the example of minimizing the cost at a single station. The horizon is $T = 10$ time periods, the initial inventory is $x_0 = 150$, with an ordering cost per unit $c = 1$, a holding cost $h = 2$ and a shortage cost $p = 3$, in appropriate measurement units. There is no fixed ordering cost. The stochastic demand is i.i.d. with mean $\bar{w} = 100$. In the robust framework, we consider that the demand belongs to the interval $[0, 200]$, that is $\hat{w} = 100$. We compare the expected costs of the robust policy and of the stochastic policy obtained by dynamic programming as a function of the standard deviation σ of the distribution.

To select the budgets of uncertainty we minimize a tight upper bound on the expected cost of the system over the set of nonnegative distributions of given mean and variance, using the results in [4]. This yields for all k :

$$\Gamma_k = \min \left(\frac{\sigma}{\hat{w}} \sqrt{\frac{k+1}{1-\alpha^2}}, k+1 \right), \quad (27)$$

and the modified demand at time k is in this example:

$$w'_k = \bar{w} + \frac{\alpha \sigma}{\sqrt{1-\alpha^2}} (\sqrt{k+1} - \sqrt{k}), \quad (28)$$

with $\alpha = \frac{p-h}{p+h}$. Expected costs are computed using the mean of a sample of size 1,000.

In the first set of experiments, the stochastic policy is computed using a binomial distribution. In the second set of experiments, the stochastic policy is computed using an approximation of the gaussian distribution on seven points $(\bar{w} - 3\sigma, \bar{w} - 2\sigma, \dots, \bar{w} + 2\sigma, \bar{w} + 3\sigma)$. In both cases, the actual distribution is Gamma, Lognormal or Gaussian, with the same mean \bar{w} and standard deviation σ . The impact of the mistake on the demand distribution is measured by the ratio $(DP - ROB)/DP$, with DP , resp. ROB , the expected cost obtained using dynamic programming, resp. the robust approach. The results are shown in Fig. 2. In the first case, where the distributions are very different beyond their first moments, the impact of the ratio increases as the standard deviation increases and the robust policy outperforms dynamic programming by up to 8%. In the second case, the two methods are equivalent in terms of performance, since the robust policy outperforms dynamic programming by at most 0.3%, which is not statistically significant.

In Figs. 3-5, we study the impact of the cost parameters c , h and p in the settings described above, where we vary one parameter at a time. The impact

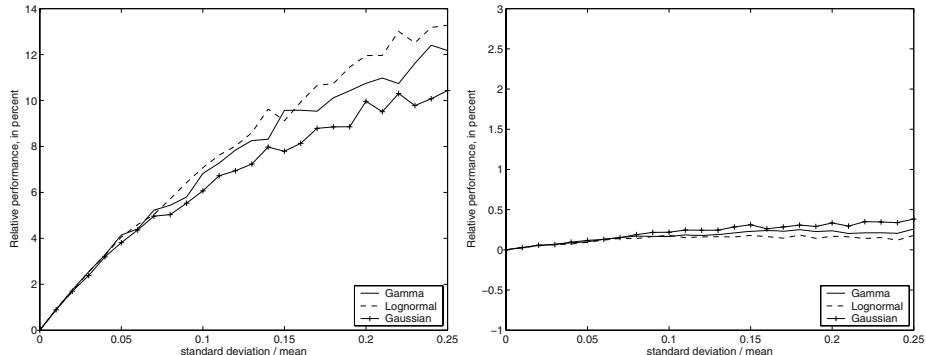


Fig. 2. Impact of the standard deviation on performance.

of a change in the parameters is qualitatively similar in both cases, with little dependence on the actual distribution of the demand. The robust approach outperforms the stochastic policy for a wide range of parameters, although the stochastic policy leads to better results for large values of the ratio p/h (greater than about 3). The exact numbers depend on the distribution used to compute the stochastic policy.

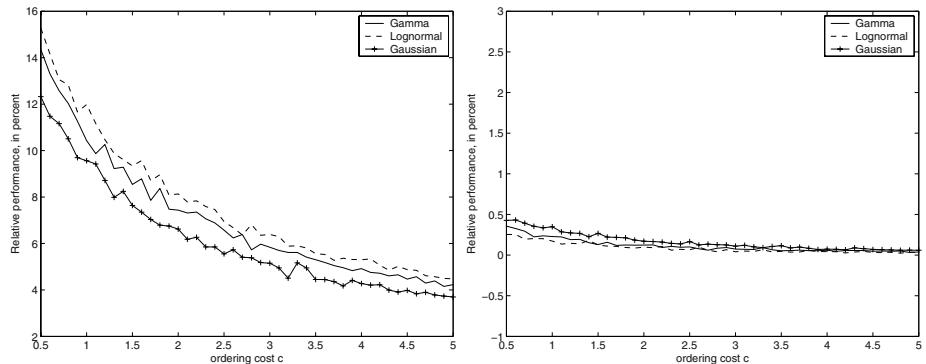


Fig. 3. Impact of the ordering cost.

Overall the numerical evidence suggests that the robust policy performs significantly better than dynamic programming when assumed and actual distributions differ widely despite having the same mean and standard deviation, and performs similarly to dynamic programming when assumed and actual distributions are close. The results are thus quite promising.

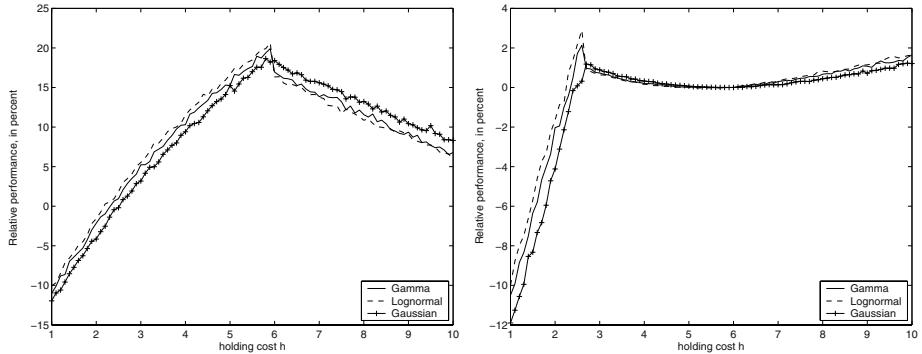


Fig. 4. Impact of the holding cost.

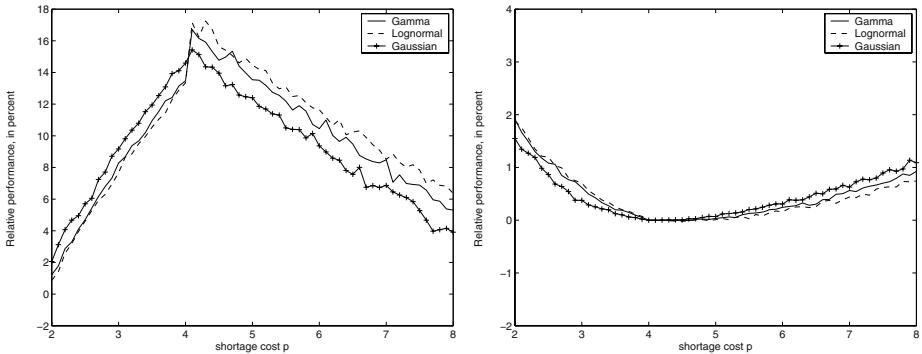


Fig. 5. Impact of the shortage cost.

6 Conclusions

We have proposed a deterministic, numerically tractable methodology to address the problem of optimally controlling supply chains subject to uncertain demand. Using robust optimization ideas, we have built an equivalent model without uncertainty of the same class as the nominal problem, with a modified demand sequence. Specifically, the proposed model is a linear programming problem if there are no fixed costs throughout the supply chain and a mixed integer programming problem if fixed costs are present.

The key attractive features of the proposed approach are: (a) It incorporates a wide variety of phenomena, including demands that are not identically distributed over time and capacity on the echelons and links; (b) it uses very

little information on the demand distributions; (c) it leads to qualitatively similar optimal policies (basestock policies) as in dynamic programming; (d) it is numerically tractable for large scale supply chain problems even in networks, where dynamic programming methods face serious dimensionality problems; (e) in preliminary computational experiments, it often outperforms dynamic programming based solutions for a wide range of parameters.

References

1. Ben-Tal, A., Nemirovski, A. (2000): Robust solutions of linear programming problems contaminated with uncertain data, *Math. Programm., Ser. A* 88:411-424.
2. Ben-Tal, A., Nemirovski, A. (1999): Robust solutions of uncertain linear programs, *Oper. Research Lett.* 25, 1-13.
3. Ben-Tal, A., Nemirovski, A. (1998): Robust convex optimization, *Math. Oper. Res.* 23, 769-805.
4. Bertsimas, D., Popescu, I. (2002): On the relation between option and stock prices: a convex optimization approach, *Oper. Res.* 50, 358-374.
5. Bertsimas, D., Sim, M. (2003): The price of robustness, to appear in *Operations Research*.
6. Bertsimas, D., Sim, M. (2003): Robust discrete optimization and network flows, *Mathematical Programming*, Vol. 98, No. 1-3, 49-71.
7. Clark, A., Scarf, H. (1960): Optimal policies for a multi-echelon inventory problem, *Management Science Vol.6 No.4*, 475-490.
8. El-Ghaoui, L., Lebret, H. (1997): Robust solutions to least-square problems to uncertain data matrices, *SIAM J. Matrix Anal. Appl.* 18, 1035-1064.
9. El-Ghaoui, L., Oustry, F., Lebret, H. (1998): Robust solutions to uncertain semidefinite programs, *SIAM J. Optim.* 9, 33-52.
10. Scarf, H. (1958): A min-max solution of an inventory problem, *Studies in the mathematical theory of inventory and production*, 201-209.
11. Soyster, A.L. (1973): Convex programming with set-inclusive constraints and applications to inexact linear programming, *Oper. Res.* 21, 1154-1157.
12. Zipkin, P. (2000): Foundations of inventory management, McGraw-Hill Higher Education.

Hedging Uncertainty: Approximation Algorithms for Stochastic Optimization Problems*

R. Ravi and Amitabh Sinha

Graduate School of Industrial Administration, Carnegie Mellon University,
Pittsburgh, PA, USA
`ravi@cmu.edu, asinha@andrew.cmu.edu`

Abstract. We study the design of approximation algorithms for stochastic combinatorial optimization problems. We formulate the problems in the framework of two-stage stochastic optimization, and provide nearly tight approximations. Our problems range from the simple (shortest path, vertex cover, bin packing) to complex (facility location, set cover), and contain representatives with different approximation ratios.

The approximation ratio of the stochastic variant of a typical problem is of the same order of magnitude as its deterministic counterpart. Furthermore, common techniques for designing approximation algorithms such as LP rounding, the primal-dual method, and the greedy algorithm, can be carefully adapted to obtain these results.

1 Introduction

With the increasing success of optimization algorithms in process optimization, these methods are making inroads into earlier planning stages of large scale projects. The inherent difference between optimization at the planning stage and *post-facto* optimization is that in the former, data is not fully available. Yet costly decisions with wide-ranging implications need to be taken in the face of incomplete data. Nevertheless, quite often forecasts of future uncertainty are available that can be used in the planning model. Forecasts, by nature, are imprecise and provide at best a range of possible futures. The field of stochastic optimization is an attempt to model such situations. For a detailed introduction, the reader is referred to one of the recent texts on the topic [4,19].

In a parallel development, the field of approximation algorithms [33,2] evolved to counter the prohibitive resource requirements for exact solution of NP-hard combinatorial optimization problems. Informally, these algorithms run in polynomial time and deliver a performance ratio on the quality of the output solution over all instances. As the size of the models being solved increases in scale, this solution approach gains in importance.

* This work was supported in part by NSF grant CCR-0105548 and ITR grant CCR-0122581 (The ALADDIN project).

However, as approximation algorithms become more sophisticated in scope and technique, the refrain from real-world practitioners who have the need for such algorithms is that the input data is seldom well-defined, thus diminishing the value of the solutions and guarantees provided by the algorithm. Conversely, while the field of stochastic optimization models the uncertainty in data fairly well, the running times of the exact algorithms developed in the stochastic optimization community often prove prohibitive. This paper combines the best of both worlds, by providing approximation algorithms for the stochastic version of several classical optimization problems.

2 Background

Two-stage Model Among the most popular models in stochastic optimization is the two-stage model with recourse. At the outset, some data may be known deterministically, whereas the uncertain future is characterized only by a probability distribution. The decisions made at this point are referred to as the first-stage decisions. Subsequently, the actual future is realized, and then there may be the opportunity to augment the first-stage solution in order to optimize for the realized scenario. This second stage of decision making is called the recourse stage. The goal is to optimize the first-stage decision variables so as to minimize the expected cost over both stages.

Mathematical Formulation We consider the two-stage stochastic optimization problem with recourse, with an additional restriction: finitely many scenarios. This means that the future will be one of a finite set of possibilities (scenarios), and the parameters and probability of occurrence of each scenario is known up-front. The mathematical formulation of this model is given below.

Vector x^0 is the set of first-stage decision variables, with constraints $Ax^0 = b$, and a cost vector is c . There are m scenarios, with the k^{th} scenario having probability of occurrence p_k , cost vector q^k , and decision variables x^k . If the k^{th} scenario is realized, then the combined solution (x^0, x^k) must satisfy the constraints given by the matrix T^k and requirement vector h^k . Let P denote additional constraints such as non-negativity or integrality.

$$\begin{aligned} \min \quad & c^T x^0 + \sum_{k=1}^m p_k (q^k)^T x^k \quad (IP_S) \\ \text{s.t.} \quad & Ax^0 = b \\ & T^k(x^0, x^k) = h^k \quad k = 1, 2, \dots, m \\ & (x^0, x^k) \in P \quad k = 1, 2, \dots, m \end{aligned}$$

The interested reader may refer to any of the texts cited above for a more complete description of models of stochastic optimization and their uses. Schultz, Stougie and van der Vlerk [30] provide an excellent survey of two-stage stochastic integer programming, while Kong and Schaefer [20] recently provided approximation algorithms for a class of such problems.

Approximation Algorithms The *raison d'être* for approximation algorithms is the prohibitively high running time of exact algorithms for integer program-

Problem	Det. approx.	Stochastic elements	Our results (Apx. ratio)	Hardness
Bin packing	APTAS [5]	Object sizes	APTAS	NP-comp. [5]
Shortest paths	1[7]	Sink only	5 $O(\log^2 n \log m)$	MAX-SNP $\Omega(\log^2 n)$
		Sink and metric		
Vertex cover	2[27]	Vertex weights, incidence	2	1.16 [14]
Facility location	1.52[25]	Demands, facility costs	8	1.46 [10]
Set cover	$O(\log n)$ [17]	Set weights, inclusions	$O(\log nm)$	$\Omega(\log n)$ [1] $\Omega(\log m)$

Fig. 1. Summary of results. We use m to denote the number of scenarios and n to refer to the number of combinatorial elements (number of vertices in graph problems and number of elements in the set cover problem).

ming and combinatorial optimization, due to their NP-completeness. Approximation algorithms run in time polynomial in the size of the input, and also provide guarantees in the form of approximation ratios. If C is a class of minimization problems and $OPT(I)$ and $A(I)$ respectively denote the value of an optimal solution and the solution output by algorithm A , then the approximation ratio $\rho(A)$ of algorithm A is defined as $\rho(A) = \max_{I \in C} \frac{A(I)}{OPT(I)}$.

While the area of approximation algorithms has been a very active field, most approximation algorithms assume complete knowledge of the input at the outset, barring a few exceptions such as scheduling problems [26,32]. Recently, and independently of us, Immorlica et al. [15] considered approximation algorithms for stochastic optimization problems with a restriction on the cost function: in the second stage, all costs go up uniformly by a factor of λ . A generalization of their model was considered by Gupta et al. [12], who provided approximation algorithms with only *sampling access* to the second-stage realization process, thereby obtaining a framework which can handle an arbitrary number of scenarios as long as there is an efficient process to sample the second stage.

Our Results We demonstrate the relevance and applicability of developing approximation algorithms for stochastic optimization. We carefully adapt existing techniques for deterministic versions of several problems to provide approximation guarantees for the stochastic versions within constant factors.

Our results are summarized in Figure 1. The current best known deterministic approximations are listed, with a “1” meaning that the problem can be solved optimally in polynomial time. All the stochastic approximation ratios are derived in this paper. Some of the hardness results are carried over from the underlying deterministic problems; the remaining are proved in this paper. An APTAS is an asymptotic polynomial time approximation scheme, which is an algorithm whose performance ratio approaches 1 as the number of objects increases. A problem is said to be MAX-SNP-hard [28], abbreviated MAX-SNP in the table, if there

is a constant $c > 1$ such that it is impossible to approximate the problem with performance ratio smaller than c unless $P = NP$.

Paper Outline In the sequel, we consider the five problems listed in Figure 1. We consider Stochastic Vertex Cover in the next section, and prove our results. Subsequent sections are devoted to the stochastic versions of four other problems that we study: Facility Location, Shortest Paths, Bin Packing and Set Cover. We conclude with directions for future research in Section 8.

3 Vertex Cover

We are given a first-stage (undirected) graph $G = (V, E_0)$, with m possible scenarios, each consisting of a probability of occurrence p_k and a set of edges E_k (not necessarily subsets of E_0). The first-stage cost of vertex v is c_v^0 , and its cost in scenario k is c_v^k . The objective is to identify a set of vertices to be selected in the first stage, so that the expected cost of extending this set to a vertex cover of the edges of the realized second-stage scenario is minimized.

We require that the edges in $E_k \setminus E_0$ have to be covered in the second stage, even if one of their end-points was chosen in the first stage. This generalizes the case when first-stage vertices cover all second-stage edges incident to them. The best known approximation algorithm for the deterministic version of vertex cover has performance ratio $2 - \frac{\log \log |V|}{2 \log |V|}$, due to Monien and Speckenmeyer [27]. A lower bound of 1.16 on the hardness of approximating the problem was shown by Håstad [14]. Our algorithm for the generalized stochastic version of vertex cover has approximation ratio 2, asymptotically matching the best known approximation for the deterministic version.

Integer Program Formulation In formulation IP_{SVC} below, variable x_v^k indicates whether or not vertex v is purchased in scenario k (where $k = 0$ denotes the first stage). Edges in $E_k \cap E_0$ may be covered in either the first or second stage, while edges in $E_k \setminus E_0$ must be covered in the second stage. Observe that a 4-approximation can be easily obtained by rounding IP_{SVC} [15]. We obtain a tighter approximation ratio using a primal-dual algorithm.

$$\begin{aligned} & \min c^0 x^0 + \sum_{k=1}^m p_k c^k x^k && (IP_{SVC}) \\ \text{s.t. } & x_u^0 + x_v^0 + x_u^k + x_v^k \geq 1 && \forall uv \in E_k \cap E_0, \forall k \\ & x_u^k + x_v^k \geq 1 && \forall uv \in E_k \setminus E_0, \forall k \\ & x && \text{non-neg. integers} \end{aligned}$$

Dual Program The dual of the linear relaxation of IP_{SVC} is shown below. Variable y_e^k packs edge e in E_k if $e \in E_k$, and it packs $e \in E_0$ if $e \in E_k \cap E_0$.

$$\begin{aligned} & \max \sum_{k=1}^m \sum_{u,v \in V} y_{uv}^k && (DP_{SVC}) \\ \text{s.t. } & \sum_{e \in E_k: v \in e} y_e^k \leq p_k c_v^k \quad \forall v, \forall k \\ & \sum_{k=1}^m \sum_{e \in E_0 \cap E_k: v \in e} y_e^k \leq c_v^0 \quad \forall v \\ & y \geq 0 \end{aligned}$$

Algorithm The algorithm is a greedy dual-ascent type of primal-dual algorithm, with two phases. In Phase I, we raise the dual variables y_e^k uniformly for all edges in $E^k \setminus E_0$, separately for each k from 1 to m . All vertices which become tight (have the first dual constraint packed to $p_k c_v^k$) have x_v^k set to 1, and are deleted along with adjacent edges. We proceed this way until all edges in $E^k \setminus E_0$ are covered and deleted.

In Phase II, we do a greedy dual-ascent on all *uncovered* edges of E_k , which are contained in $E_k \cap E_0$. We raise y_e^k for all uncovered edges for $k = 0$ to m . We use a slightly different rule for purchasing vertices: If a vertex is tight for x_v^0 (i.e., second dual constraint packed to c_v^0), then we select it in the stage 1 solution by setting $x_v^0 = 1$, and if it is not tight for x^0 but is tight for x^k (packed in the first dual constraint), then we select it in the recourse solution and set $x_v^k = 1$.

Theorem 1. *The integer program IP_{SVC} can be rounded by the primal-dual algorithm described above within a factor of 2 in polynomial time.*

Proof. Consider an edge $e = uv$ in scenario k . We must have selected one of its two end-points in either Phase I or Phase II (or both), so the algorithm yields a feasible solution. We use linear programming duality to bound the cost of the solution by showing that the cost of our solution is no more than $2 \sum_k \sum_{u,v \in V} y_{uv}^k$, where y is the dual solution constructed by our algorithm. Each time we set an x_v^k variable to 1, we assign some dual variables to it such that (i) the sum of dual variables assigned to each such x_v^k variable equals $p_k c^k$ (where $p_0 = 1$), and (ii) each dual variable is assigned at most twice.

Consider a vertex v which was selected (ie, x_v^k was set to 1) in scenario k in either Phase I or Phase II. We assign all dual variables y_e^k such that v is an end point of e to this vertex, and since v is selected only when the constraint $\sum_{e \in E_k: v \in e} y_e^k \leq p_k c_v^k$ goes tight, we maintain (i). An edge e in $E_k \setminus E_0$ is assigned to a vertex v only if x_v^k is set to 1 for $k \neq 0$, and since an edge has at most 2 end-points, we ensure (ii) for edges in E_k .

Next consider a vertex v for which we set x_v^0 to 1. Therefore, the constraint $\sum_l \sum_{e \in E_0 \cap E_k: v \in e} y_e^k \leq c_v^0$ must have gone tight, and all edges in the sum are assigned to the variable x_v^0 . This assignment once again ensures (i). This assignment only includes edges in $E_0 \cap E_k$, and these edges are not assigned to any variable x_v^k for $k \neq 0$, thus also ensuring (ii) for all edges in $E_0 \cap E_k$.

These two cases cover all the possibilities, thus proving the theorem.

4 Facility Location

As in the classical uncapacitated facility location problem, we are given a set of facilities F and a set of clients D , with a metric c_{ij} specifying the distances between every client and every facility. However, the demand of each client is not known at the first stage. In scenario k , client j has demand d_j^k , which may be zero. Facility i has a first-stage opening cost of f_i^0 , and recourse costs of f_i^k in scenario k . These may be infinity, reflecting the unavailability of the facilities in various scenarios. We abbreviate this problem as SFL .

$$\begin{aligned}
& \min \sum_{i \in F} f_i y_i^0 + \sum_{k=1}^m p_k \left(\sum_{i \in F} f_i^k y_i^k + \sum_{i \in F, j \in D} d_j^k c_{ij} x_{ij}^k \right) (IP_{SFL}) \\
\text{s.t. } & \sum_{i \in F} x_{ij}^k \geq d_j^k & \forall j \in D, \forall k \\
& x_{ij}^k \leq y_i^0 + y_i^k & \forall i \in F, \forall j \in D, \forall k \\
& x, y \quad \text{non-negative integers}
\end{aligned}$$

The problem is best explained by the IP formulation IP_{SFL} above. While our algorithm extends to arbitrary demands, for simplicity we only study the case when all d_j^k 's are either 0 or 1. Variable x_{ij}^k is 1 if and only if client j is served by facility i in scenario k . If $x_{ij}^k = 1$, then facility i must either be opened at the first stage ($y_i^0 = 1$) or in recourse in scenario k ($y_i^k = 1$) (or both).

History and Non-triviality of the Problem The classical (deterministic) uncapacitated facility location problem has a rich history (see Cornuéjols, Nemhauser and Wolsey [6] for a survey). Balinski [3] introduced an integer programming formulation for this problem which has led to several approximation algorithms. The first constant factor approximation using this formulation is due to Shmoys, Tardos and Aardal [31], and the current best algorithm, due to Mahdian, Ye and Zhang [25] uses a formulation which differs only slightly. Indeed, our formulation (IP_{SFL}) extends Balinski's formulation to the stochastic setting. In the stochastic optimization community, Louveaux and Peeters [23] considered a slightly different version of stochastic facility location, and provided a dual-ascent based exact (non-polynomial time) algorithm for it.

Notice that if the second stage facility costs were identical to those in the first stage for all scenarios, then we can “de-couple” the stochastic components of the problem and solve for each scenario independently. On the other hand, if there was no second stage and all facilities had to be opened in the first stage, then SFL reduces to an instance of the usual UFL, where the probability multipliers in the expected service costs can be incorporated into the demand terms (thinking of the demands as being scaled based on the probability of occurrence); in this case, existing approximations for UFL apply directly.

The added difficulty, and indeed the interesting aspect of the model, arises from varying (and typically increased) second-stage facility costs under different scenarios. We cannot treat each scenario by itself, since the different scenarios interact in utilizing first-stage facilities. Our algorithm has to carefully balance the effects of the two stages in deciding what facilities to open.

Algorithm Our approximation algorithm proceeds along the lines of the LP-rounding algorithm of Shmoys, Tardos and Aardal [31], with some crucial differences. We begin by solving the linear relaxation of IP_{SFL} . Let (x, y) denote an optimal LP solution. The first step in rounding this fractional solution is using the filtering technique of Lin and Vitter [22]. We fix a constant $0 < \alpha < 1$. For every client-scenario pair (j, k) , we define its optimal fractional service cost to be $c_{jk}^* = \sum_i c_{ij} x_{ij}^k$. Order the facilities which serve the pair (j, k) according to non-decreasing distance from j . The α point $g_{j,k}(\alpha)$ for the client-scenario pair

(j, k) is the smallest distance c_{jk}^α such that $\sum_{i:c_{ij} \leq c_{jk}^\alpha} x_{ij}^k \geq \alpha$. The following theorem was also proved in [31].

Theorem 2. *Given a feasible fractional solution (x, y) , we can find a fractional solution (\bar{x}, \bar{y}) which is feasible for the LP relaxation of IP_{SFL} in polynomial time such that (i) $c_{jk}^\alpha \leq \frac{1}{1-\alpha} c_{jk}^*$; (ii) $\bar{x}_{ij}^k > 0 \Rightarrow c_{ij} \leq c_{jk}^\alpha$ for all $i \in \mathcal{F}, j \in \mathcal{D}, k = 1, 2, \dots, m$; (iii) $\bar{y}_i^k \leq \min\{1, \frac{y_i^k}{\alpha}\}$ for all $i \in \mathcal{F}, k = 0, 1, \dots, m$.*

Proof. First, if $c_{jk}^\alpha > \frac{1}{1-\alpha} c_{jk}^*$, then we get the following contradiction (as an application of Markov's inequality): $c_{jk}^* \geq \alpha \cdot 0 + (1 - \alpha)c_{jk}^\alpha > c_{jk}^*$, proving (i).

Next, define \bar{x} as follows, which satisfies (ii) by definition:

$$\bar{x}_{ij}^k = \begin{cases} \min\{1, \frac{1}{\alpha}\} x_{ij}^k & \text{if } c_{ij} \leq c_{jk}^\alpha \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, define $\bar{y}_i^k = \min_{j \in \mathcal{D}} \bar{x}_{ij}^k$. Using (i) and the definition of \bar{x} , it follows that $\bar{y}_i^k \leq \min\{1, \frac{y_i^k}{\alpha}\}$ for all $i \in \mathcal{F}$, satisfying (iii). The definitions also ensure that (\bar{x}, \bar{y}) is a feasible solution to the LP relaxation of IP_{SFL} .

The algorithm of Shmoys, Tardos and Aardal [31] iteratively rounds x_{ij}^k variables for which c_{jk}^α is smallest. This does not work in our case, because the rounding algorithm might close facilities which are needed for other scenarios $k' \neq k$. Hence we need a rounding algorithm which carefully treats the distinction between stage 1 facility variables y^0 , and recourse facility variables y^k .

We proceed as in earlier algorithms by obtaining an optimal LP solution; In the next step, we progressively choose clients *across all scenarios* with minimum fractional service cost, and neglect to serve other clients conflicting (overlapping in facility utilization) with it by assigning them to be served by this client's serving facility. The main difference is that if a stage 1 facility is opened to serve a client, all clients that conflict with it can be served, while if a stage 2 facility variable is rounded up to serve this client, only those clients in the same scenario that conflict with this client are neglected and assigned to this client. This strategy suffices to pay for all opened facilities by the “disjointness” of the different scenarios’ contributions in the objective function, while the rule of considering clients in increasing order of fractional service cost allows us to bound the service cost. Our rounding algorithm is described in detail below. Let $0 < \beta < 1$ be another fixed constant.

1. Initialize $\hat{F}^k = \emptyset$ to be the set of facilities opened in scenario k for $k = 0, 1, \dots, m$. Mark all client-scenario pairs as “unserved”.
2. Let (j, k) be an unserved client-scenario pair with smallest c_{jk}^α . Consider the following cases, in each case marking (j, k) as “served” and proceeding to the next client-scenario pair. Let S^0 be the set of facilities i such that $\bar{x}_{ij}^k > 0 \wedge \bar{y}_i^0 > 0$, and S^k be the set of facilities i such that $\bar{x}_{ij}^k > 0 \wedge \bar{y}_i^k > 0$.
 - (a) If $\sum_{i \in S^0} \bar{y}_i^0 \geq \beta$, let i be the facility such that f_i^0 is smallest among all facilities in S^0 . Move facility i to the set \hat{F}^0 , and set $\hat{y}_i^0 = 1$. For all

other facilities $i' \in S^0 \cup S^k$, set $\hat{y}_{i'}^0 = \hat{y}_{i'}^k = 0$. For client-scenario pairs (j', k') such that there exists a facility $i' \in S^0 \cup S^k$ with $c_{i'j'} \leq c_{j'k'}^\alpha$, set $\hat{x}_{i'j'}^{k'} = 1$ and mark them as “served”.

- (b) If $\sum_{i:i \in S^0} \bar{y}_i^0 < \beta$, then we must have $\sum_{i:c_{ij} \leq c_{jk}^\alpha} \bar{y}_i^k \geq 1 - \beta$. In this case, let i be the facility in S^k with smallest f_i^k . Move facility i to the set \hat{F}^k and set $\hat{y}_i^k = 1$. For all other facilities $i' \in S^k$, set $\hat{y}_{i'}^k = 0$. For clients j' such that there exists a facility $i' \in S^k$ with $c_{i'j'} \leq c_{j'k}^\alpha$, set $\hat{x}_{i'j'}^k = 1$ and mark them as “served”.
- 3. Facilities in \hat{F}^0 are the facilities to be opened in stage 1, and facilities in \hat{F}^k are the facilities to be opened in recourse if scenario k materializes. Clients are served according to the zero-one variables \hat{x}_{ij}^k .

Lemma 1. *The rounding algorithm above produces an integer solution (\hat{x}, \hat{y}) which is feasible for IP_{SFL} such that (i) For every client-scenario pair (j, k) , we have $\hat{x}_{ij}^k = 1 \Rightarrow c_{ij} \leq 3c_{jk}^\alpha$. (ii) $\sum_{i \in F} f_i^0 \hat{y}_i^0 \leq \frac{1}{\beta} \sum_{i \in F} f_i^0 \bar{y}_i^0$. (iii) $\sum_{i \in F} f_i^k \hat{y}_i^k \leq \frac{1}{1-\beta} \sum_{i \in F} f_i^k \bar{y}_i^k$ for all $k = 1, 2, \dots, m$.*

Proof. When a client is assigned to a facility (ie, \hat{x}_{ij}^k is set to 1), we either assign it to a facility within distance c_{jk}^α , or it is assigned when some other client j' with $c_{j'k}^\alpha \leq c_{jk}^\alpha$ was being considered. In either case, a simple application of triangle inequality yields $c_{ij} \leq 3c_{jk}^\alpha$.

When a facility i is chosen for opening in the first stage (ie, \hat{y}_i^0 is set to 1), case 2(a) must have occurred. In that case, we have a sufficiently large fraction (β) of facilities which have $\bar{y}_i^0 > 0$ which we are shutting, and we can charge the cost of opening i to the fractional solution. A similar argument holds for the case when a facility is opened in recourse in scenario k .

The solution produced is also feasible, because we start with a feasible solution (\bar{x}, \bar{y}) , and in each step, we maintain feasibility by ensuring that a client-scenario pair is marked “served” only when its x_{ij}^k variable is set to 1 (ie, it is assigned to a facility) for some facility i .

Theorem 3. *There is a polynomial time approximation algorithm with performance ratio 8 for SFL.*

Proof. Setting $\alpha = \frac{1}{4}$ and $\beta = \frac{1}{2}$, along with Theorem 2 and Lemma 1, yields the performance guarantee.

Extensions The algorithm easily extends to allowing demands at client-scenario pairs which are non-negative real numbers instead of just 0 or 1. We may also allow the costs to transport one unit of demand per unit length in different scenarios to be different. In other words, each scenario has a multiplier γ_k such that the distance between i and j in scenario k is $\gamma_k c_{ij}$. Essentially, this can be incorporated into the demand variables d_j^k . Recently, Mahdian [24] developed an approximation algorithm for SFL with approximation ratio 3, by extending the ideas of Jain and Vazirani [16].

5 Shortest Paths

Motivation Consider a supplier who wishes to ship a single unit of a good to a single destination t from a single source s , in a graph where the shipping cost is just the cost of the edge. The solution to this problem is to compute a shortest path from s to t , and this can be easily done in polynomial time, for example by using the algorithm due to Dijkstra [7].

5.1 Stochastic Sink

Now consider the case when the supplier does not know the destination in advance. In particular, any of m scenarios could materialize, with the destination being t^k in scenario k . The supplier wishes to reserve some edges now at cost c_e , and augment the network in the second stage (when edges may be more expensive) after the revelation of the actual destination.

Problem Definition We are given a graph $G = (V, E)$, with metric edge costs c_e and a single source $s \in V$. We also have a set of m scenarios, with scenario k specified by a destination vertex $t_k \in V$, a cost scale factor f_k , and a probability p_k . A feasible solution is specified by a set of edges $E' \subset E$. The first-stage cost of this solution is $\sum_{e \in E'} c_e$, and in scenario k , a second stage solution is a path P_k from s to t_k ; for the second stage costs, we assume the edges in P_k bought in the first stage, namely in E' , have cost zero, while the remaining edges are increased in cost by factor f_k , giving second-stage cost $f_k \sum_{e \in P_k \setminus E'} c_e$. The objective is to compute E' which minimizes the sum of first stage edge costs and expected second stage edge costs. We abbreviate this problem as SSP (stochastic shortest paths). While it is not obvious that E' even induces a connected component, the following lemma proves that E' is indeed connected; in fact, it is a tree.

Lemma 2. *The set of edges E' bought in the first stage in an optimal solution to SSP induces a tree containing the source s .*

Proof. Suppose for a contradiction there is another connected component $C \not\ni s$. Let K' be the set of scenarios for which the optimal solution uses at least one edge in C , and let E_s be the connected component of first-stage edges which include the source s . For optimality, it must be the case that for every edge $e \in C$, we have $\sum_{P_k \ni e} p_k f_k \geq 1$, implying that $\sum_{k \in K'} f_k \geq 1$.

Now consider the paths used in the scenarios in K' . Let k^0 be the scenario in which the second-stage cost of the segment from C to the source is the cheapest. If we re-route the paths of all scenarios in K' to using the path to using the path of k^0 from the point the other scenario paths intersect C , then since $\sum_{k \in K'} f_k \geq 1$, the total cost cannot increase. Therefore, we can purchase these edges (which we used for re-routing), and this does not increase the cost.

Proceeding this way for other components, we infer that E^* induces a connected graph containing s , which need be no more than a tree since the second stage solutions only look for a single path to s .

Interpretation as a Network Design Problem Armed with the above lemma, SSP can be interpreted as the tree-star network design problem, defined as follows. In tree-star network design, demand nodes have a demand for d_j units of goods to be shipped to a source. A feasible solution is specified by a tree, with the cost of the solution being M times the cost of the tree (for pre-specified M) plus the length of the shortest path from each demand node to the tree, weighted by the demand at the node. A constant-factor approximation algorithm for this problem was first provided by Ravi and Salman [29], and it has also been studied subsequently as the connected facility location problem [18,21], and the asymmetric VPN design problem [11].

Theorem 4. *There is a polynomial-time constant-factor approximation algorithm for SSP.*

Proof. SSP is equivalent to the tree-star network design problem, via the following transformation. The fixed cost multiplier of the tree M is set to 1. The demand of each node t_k is set to $f_k p_k$. Now purchasing a tree T in stage 1 for SSP is equivalent to building T in the tree-star problem. The expected second stage cost is exactly $\sum_{k=1}^m p_k f_k \text{dist}(t_k, T)$, which is the same as incurred in the tree-star problem when the demand at node t_k is $p_k f_k$.

The equivalence of SSP and tree-star network design also implies the NP-hardness of SSP. The best-known approximation ratio for tree-star network design is 5, due to Kumar and Swamy [21]. This implies an approximation algorithm with the same performance ratio for stochastic sink shortest paths.

5.2 Stochastic Metric and Sink

The problem becomes even more interesting (and harder) when the metric itself is allowed to change arbitrarily across scenarios. This might happen, for example, because shipping by sea becomes much cheaper than air transport in one scenario, and vice-versa in another. The problem is defined exactly as in Section 5, except that the cost of edge e in the first stage is c_e^0 and in scenario k is c_e^k . We call this the stochastic metric shortest paths (SMSP) problem.

In general, the first-stage component of an optimal solution for SMSP need not be a tree. Consider the following example, where there is only one second-stage scenario. The graph is a path with five vertices $s = v_0, \dots, v_4 = t$, where s and t are the source and the sink respectively. Let M be a large constant. The costs of the four edges $(v_0, v_1), \dots, (v_3, v_4)$ in the first stage are respectively 1, M , 1, M , and in the second stage are M , 1, M , 1. The optimal solution is clearly to purchase edges (v_0, v_1) and (v_2, v_3) in the first stage, and the others in the second stage; this solution has cost 4. Any solution which requires the first stage to be a tree has cost at least M .

Hardness Even with the restriction that the first stage set of edges form a tree, SMSP is as hard as the group Steiner tree problem (GST), defined as follows. $G = (V, E)$ is an undirected graph with edge weights c_e , and there are

m vertex subsets (called groups) S_k . The objective is to compute a minimum cost tree which includes at least one vertex from every group. This problem was studied by Garg, Konjevod and Ravi [9] who also gave an approximation algorithm with performance ratio roughly $O(\log^2 n \log m)$, and recently Halperin and Krauthgamer [13] showed an inapproximability threshold of $\Omega(\log^2 n)$ even when G is a tree. For the rest of this section, we consider the restriction of SMSP where the first stage solution has to be a tree, which we dub *Tree-SMSP*. An $\Omega(\log^2 n)$ hardness for Tree-SMSP follows from the reduction of GST to Tree-SMSP, shown below.

Theorem 5. *A GST instance can be modeled as a special case of Tree-SMSP.*

Proof. Suppose we are given an instance of group Steiner tree, specified by $G = (V, E)$, metric edge costs c , and groups S_1, S_2, \dots, S_m . We create an instance of SMSP with one scenario for every group. The graph remains the same, and the first stage edge costs c^0 are the same as c , the edge costs in the GST instance. In scenario k , the metric is as follows. The distance between any two vertices in S_k is zero, and all other distances are infinity. Any vertex in S_k is defined to be the destination t_k for scenario k . All scenarios are equally likely.

An optimal solution to this instance of Tree-SMSP must select a first stage tree which includes at least one vertex from each S_k , to avoid infinite cost. If the tree includes any vertex in S_k , it can be augmented at cost zero to a tree which includes t_k if scenario k materializes.

Approximation Algorithm Our approximation algorithm relies on the following IP formulation of Tree-SMSP. Variable r_{uv}^k is 1 if edge (u, v) (in the direction $u \rightarrow v$) is part of the path traversed from t_k to s and edge (u, v) is chosen in the recourse solution. Variable f_{uv}^k is 1 if edge (u, v) is chosen in the path from t_k to s and edge (u, v) is part of the first-stage solution. Variable x_{uv} is 1 if edge (u, v) is chosen in the first-stage tree.

$$\begin{aligned} & \min \sum_e c_e x_e + \sum_{k=1}^m p_k \sum_e r_e^k c_e^k \quad (IP_{SMSP}) \\ \text{s.t. } & \sum_v (r_{t_k, v}^k + f_{t_k, v}^k) \geq 1 \quad \forall k \\ & \sum_v (r_{uv}^k + f_{uv}^k) = \sum_v (r_{vu}^k + f_{vu}^k) \quad \forall u \in V \setminus \{t_k, s\}, \forall k \\ & \sum_v r_{uv}^k \leq \sum_v r_{vu}^k \quad \forall u \in V \setminus \{t_k\}, \forall k \\ & f_e^k \leq x_e \quad \forall e \in E, \quad \forall k \\ & f, r, x \quad \text{non-neg. integers} \end{aligned}$$

The third set of inequalities are strengthenings valid only for the tree version of SMSP, insisting that flows along recourse arcs from t_k to s via any node are non-increasing; they are also crucial for obtaining the result below. IP_{SMSP} is polynomial in size, so its linear relaxation LP_{SMSP} can be solved optimally in polynomial time. Let (f, r, x) denote an optimal solution to the linear program LP_{SMSP} , and OPT_{SMSP} be its value. The following theorem describes our rounding algorithm.

Theorem 6. *The fractional solution (f, r, x) can be rounded in polynomial time to an integer solution $(\hat{f}, \hat{r}, \hat{x})$ of cost $O(\log^2 n \log m) OPT_{Tree-SMSP}$.*

Proof. For each destination t_k , let $r^*(k) = \sum_e r_e^k c_e^k$ be the cost incurred by the recourse component of the fractional path for t_k . Let S_k be the set of all nodes within distance $2r^*(k)$ of t_k in the metric c^k . The idea is that we can incur a factor of 2 and pay for a path from t_k to any node in S_k by charging it to $r^*(k)$, and hence we need a first stage tree which reaches at least one node in S_k . We construct sets S_k for every scenario k , and create an instance of the group Steiner tree problem using the metric c .

Using Markov's inequality, if $s \notin S_k$, we have $\sum_{e=(u,v):u \in S_k, v \notin S_k} x_e \geq \frac{1}{2}$. Hence $2x$ is a solution to the LP relaxation of the following IP formulation of the group Steiner tree problem: $\min \sum_e c_e x_e$ such that $\sum_{e=(u,v):u \in S, v \notin S} x_e \geq 1 \forall S \exists k : S_k \subseteq S$. Using the result of Garg, Konjevod and Ravi [9], we can construct an integer tree solution \hat{x} at a cost $O(\log^2 n \log m \cdot OPT_{SMSP})$ which includes at least one vertex of every S_k . Since for every scenario k we can augment this tree to include t_k at cost at most $2r^*(k)$, our approximation ratio follows.

6 Stochastic Bin Packing

Stochastic bin packing is motivated by applications where storage capacity has to be reserved in advance of the arrival of the objects, and if the reserved capacity is insufficient, we have to purchase additional capacity at possibly higher costs. Formally, we are given a bin capacity B , known in advance. There is a set of m possible scenarios, with scenario k specified by a probability p_k of occurrence, a set S_k of objects (each with size $s_i^k \leq B$), and a bin cost f_k . A feasible solution is specified by a number x of bins purchased in stage 1, at unit cost per bin. If scenario k materializes, the objects in S_k need to be packed into bins of capacity B , which may necessitate the purchase of an additional number of bins at cost f_k per bin. The objective is to compute x so as to minimize the expected total cost. Let $[x]$ denote the integer nearest to x .

Let ρ denote the approximation ratio of the best approximation algorithm for the bin-packing problem. Any locally optimal algorithm (first-fit, for example) achieves $\rho = 2$. An asymptotic PTAS was given by Fernandez de la Vega and Lueker [8], which uses at most $(1+2\epsilon)OPT + 1$ bins. The following theorem shows how to extend any bin-packing algorithm to handle stochastic bin-packing.

Theorem 7. *Order the scenarios so that we have $\sum_i s_i^1 \geq \sum_i s_i^2 \geq \dots \geq \sum_i s_i^m$. Let k^* be the largest integer such that $\sum_{k=1}^{k^*} f_k p_k \geq 1$. Then $x = [\rho \sum_i s_i^{k^*}]$ is an asymptotic ρ -approximate solution.*

Proof. Consider the fractional relaxation of the problem, when we can pack items fractionally into bins. In that case, $x^* = [\sum_i s_i^{k^*}]$ is the optimal solution, because it is the point where the expected marginal cost of buying an additional bin in recourse goes below 1. The expected total cost if we purchase x^* bins

is $x^* + \sum_{k>k^*} p_k f_k(\lceil \sum_i s_i^k \rceil - x^*)$, which is a lower bound on the value of an optimal solution of stochastic bin packing.

Since $\lceil \rho \sum_i s_i^k \rceil$ bins are asymptotically sufficient to pack the objects in S_k , we will need to purchase at most $\lceil \rho \sum_i s_i^k \rceil - \rho x^*$ additional bins if scenario $k > k^*$ materializes. If scenario $k \leq k^*$ is realized, then ρx^* bins are sufficient and no additional bins are needed. Hence the expected cost of our solution is $\rho x^* + \sum_{k>k^*} p_k f_k(\lceil \rho \sum_i s_i^k \rceil - \rho x^*)$, which is asymptotically no more than ρ times our lower bound.

7 Stochastic Set Cover

The input in the stochastic set cover problem consists of a universe U of $|U| = n$ elements, and a collection \mathcal{S} of subsets of U . Each set $S \in \mathcal{S}$ has a stage 1 cost c_S^0 and a cost of c_S^k in scenario k , some of which might be infinity. Each element $u \in U$ has a demand vector d_u with the k^{th} component d_u^k being 1 if it is required to cover u in scenario k , and 0 otherwise. A feasible solution is a collection $\mathcal{S}' \subseteq \mathcal{S}$, with stage 1 cost $\sum_{S \in \mathcal{S}'} c_S^0$. If scenario k is realized, then \mathcal{S}' must be extended by with some more sets \mathcal{S}^k to cover all elements with $d_u^k = 1$. The cost of this recourse solution is $\sum_{S \in \mathcal{S}^k} c_S^k$, incurred with probability p_k .

Reduction to Classical Set Cover The deterministic version of set cover was among the earliest NP-hard problems to be approximated, with a $O(\log n)$ approximation was first provided by Johnson [17]. The problem was also shown to be NP-hard to approximate better than a factor of $\Omega(\log n)$ by Arora and Sudan [1]. Given an instance of deterministic set cover, we can define an instance of stochastic set cover by creating a distinct scenario for each element, and setting all second-stage set costs to infinity. This implies an inapproximability threshold of $\Omega(\log m)$ for stochastic set cover too.

We show below that any instance of stochastic set cover with n elements can be transformed to an instance of deterministic set cover with $n(m+1)$ elements. This means that there exists an $O(\log nm) = O(\log n + \log m)$ approximation for stochastic set cover. The approximation ratio therefore matches the inapproximability ratio upto constants. The reduction in Theorem 8 allows us to extend the model to the following generalization, for which the same approximation guarantee holds: In scenario k , each set S_k covers only a subset of the elements that the first-stage set S covers.

Theorem 8. *Any stochastic set cover problem is equivalent to a classical set cover problem with mn elements and $|\mathcal{S}|(m+1)$ sets.*

Proof. Associate an element u_k for every element-scenario pair (u, k) such that $d_u^k = 1$. Create $m+1$ copies of every set $S \in \mathcal{S}$. Set S^0 contains all elements u_k for all $k = 1, 2, \dots, m$ such that $u \in S$, while set S^k only contains u_k for all $u \in S$. Finally, the cost of S^0 is c_S^0 and that of S^k is $p_k c_S^k$. By construction, any solution to the stochastic set cover instance yields a solution to the transformed deterministic instance, and vice-versa.

8 Directions for Further Research

Much remains to be done on several classical problems for which algorithms in the two-stage stochastic model are not known. Another direction of research is to develop approximations for more complex models of stochastic optimization: The extension of the two-stage model to multiple stages allows more detailed modeling; the variant where uncertainty is modeled by a continuous distribution is also often considered. It is our hope that these models will provide a rich setting for the application of optimization in practice.

9 Acknowledgments

We would like to thank Nan Kong and Andrew Schaefer of the University of Pittsburgh for several enlightening discussions leading to this work.

References

1. Arora, S., Sudan, M. Improved low degree testing and its applications. In Proceedings of the 29th Annual ACM Symposium on Theory of Computing (1997) 485-495.
2. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties, Springer, Berlin, Germany (1999).
3. Balinski, M.L.: On finding integer solutions to linear programs. In Proc. IBM Scientific Computing Symposium on Combinatorial Problems (1966) 225-248.
4. Birge, J., Louveaux, F.: Introduction to Stochastic Programming, Springer, Berlin (1997).
5. Coffman Jr., E., Garey, M., Johnson, D.: Approximation algorithms for bin-packing: a survey. In D.S. Hochbaum, Approximation Algorithms for NP-hard Problems, PWS, Boston (1997).
6. Cornuéjols, G., Nemhauser, G., Wolsey, L.: The uncapacitated facility location problem. In P. Mirchandani and R. Francis, eds, Discrete Location Theory, Wiley, New York (1990) 119-171.
7. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische Mathematik **1** (1959) 269-271.
8. Fernandez de la Vega, W., Lueker, G.S.: Bin packing can be solved within $1 + \epsilon$ in linear time. Combinatorica **1** (1981) 349-355.
9. Garg, N., Konjevod, G., Ravi, R.: A polylogarithmic approximation algorithm for the group Steiner tree problem. Journal of Algorithms **37(1)** (2000) 66-84.
10. Guha, S., Khuller, S.: Greedy strikes back: Improved facility location algorithms. In Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (1998) 649-657.
11. Gupta, A., Kleinberg, J., Kumar, A., Rastogi, R., Yener, B.: Provisioning a virtual private network: A network design problem for multicommodity flow. In Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (2001) 389-398.
12. Gupta, A., Pál, M., Ravi, R., Sinha, A.: Boosted sampling: Approximation algorithms for stochastic optimization. Proceedings of the 36th, Annual ACM Symposium on Theory of Computing (2004) (to appear).

13. Halperin, E., Krauthgamer, R.: Polylogarithmic inapproximability. In Proceedings of the 35rd Annual ACM Symposium on Theory of Computing (2003) 585-594.
14. Hästads, J.: Some optimal inapproximability results. In Proceedings of the 29th Annual ACM Symposium on Theory of Computing (1997) 1-10.
15. Immorlica, N., Karger, D., Minkoff, M., Mirrokni, V.: On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (2004) 684-693.
16. Jain, K., Vazirani, V.: Primal-dual approximation algorithms for metric facility location and k -median problems. In Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (1999) 2-13.
17. Johnson, D.: Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences **9** (1974) 256-278.
18. Karger, D., Minkoff, M.: Building Steiner trees with incomplete global knowledge. In Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (2000) 613-623.
19. Klein Haneveld, W.K., van der Vlerk, M.H.: Stochastic Programming, Dept. of Econometrics and OR, University of Groningen, Netherlands (2003).
20. Kong, N., Schaefer, A.: A factor $\frac{1}{2}$ approximation algorithm for a class of two-stage stochastic mixed-integer programs. Manuscript, submitted to INFORMS Journal of Computing (2003).
21. Kumar, A., Swamy, C.: Primal-dual algorithms for connected facility location problems. In Approximation Algorithms for Combinatorial Optimization (2002) 256-270.
22. Lin, J.-H., Vitter, J.: ϵ -approximations with minimum packing constraint violation. In Proceedings of the 24th Annual ACM Symposium on Theory of Computing (1992) 771-782.
23. Louveaux, F., Peeters, D.: A dual-based procedure for stochastic facility location. Operations Research **40** (1992) 564-573.
24. Mahdian, M.: Personal communication (2003).
25. Mahdian, M., Ye, Y., Zhang, J.: A 1.52 approximation algorithm for the uncapacitated facility location problem. In Approximation Algorithms for Combinatorial Optimization (2002) 229-242.
26. Möhring, R., Schulz, A., Uetz, M.: Approximation in stochastic scheduling: The power of LP-based priority policies. Journal of the ACM **46(6)** (1999) 924-942.
27. Monien, B., Speckenmeyer, E.: Ramsey numbers and an approximation algorithm for the vertex cover problem. Acta Informatica **22** (1985) 115-123.
28. Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation, and complexity classes. Journal of Computer Systems and Sciences **43** (1991) 425-440.
29. Ravi, R., F.S. Salman, F.S.: Approximation algorithms for the traveling purchaser problem and its variants in network design. In European Symposium on Algorithms (1999) 29-40.
30. Schultz, R., Stougie, L., van der Vlerk, M.H.: Two-stage stochastic integer programming: A survey. Statist. Neerlandica **50(3)** (1996) 404-416.
31. Shmoys, D., Tardos, E., Aardal, K.: Approximation algorithms for facility location problems. In Proceedings of the 29th ACM Symposium on Theory of Computing (1997) 265-274.
32. Skutella, M., Uetz, M.: Scheduling precedence-constrained jobs with stochastic processing times on parallel machines. In Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (2001) 589-590.
33. Vazirani, V.: Approximation Algorithms, Springer, Berlin, Germany (2001).

Scheduling an Industrial Production Facility

Eyjolfur Asgeirsson^{1*}, Jonathan Berry^{2**}, Cynthia A. Phillips^{3***},
David J. Phillips^{1†}, Cliff Stein^{1‡}, and Joel Wein^{4,5§}

¹ Department of IEOR, Columbia University, New York, NY.
`ea367@columbia.edu`, `djp80@columbia.edu`, `cliff@ieor.columbia.edu`

² Department of Computer Science, Lafayette College, Easton, PA.
`berryjw@cs.lafayette.edu`

³ Algorithms and Discrete Mathematics Department, Sandia National Labs,
Albuquerque, NM.
`caphill@sandia.gov`

⁴ Akamai Technologies, 8 Cambridge Center, Cambridge MA 02139

⁵ Department of Computer Science, Polytechnic University, Brooklyn, NY.
`wein@mem.poly.edu`

Abstract. Managing an industrial production facility requires carefully allocating limited resources, and gives rise to large, potentially complicated scheduling problems. In this paper we consider a specific instance of such a problem: planning efficient utilization of the facilities and technicians that maintain the United States nuclear stockpile. A detailed study of this problem yields a complicated mixed-integer programming (MIP) model with upward of hundreds of thousands of variables and even more constraints. Consistently and quickly solving such a model exactly is impossible using today's algorithms and computers, and, in addition to branch-and-bound, requires good heuristics and approximation algorithms. In an effort to design such algorithms, we study several different methods of generating good solutions given the solution to the LP relaxation. We design a suite of sample data and test the algorithms. The goals of this project were twofold. First, we wanted to develop a program that could efficiently and accurately help with the Pantex planning problem. Second, we wanted to experimentally test various ideas, designed originally for "cleaner" problems, in this more challenging context. In summary, we demonstrate the value of using α -points as a way to quickly and cheaply generate, from one solution of an LP relaxation,

* Research partially supported by NSF Grant DMI-9970063 and an American-Scandinavian Foundation Thor Thors Memorial Fund Grant.

** Some of this work done while this author visited the third author at Sandia National Labs.

*** Sandia is a multipurpose laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

† Research partially supported by NSF Grants DGE-0086390 and DMI-9970063.

‡ Research partially supported by NSF Grant DMI-9970063 and an Alfred P. Sloan Foundation Fellowship.

§ Research partially supported by NSF Grant DMI-9970063.

many feasible solutions to an integer program. In this particular environment, the use of α -points, combined with other heuristics, outperforms local search. We also see the value of finding combinatorially-structured subproblems as opposed to using simple greedy approaches.

1 Introduction

Managing an industrial production facility requires carefully allocating limited resources, and gives rise to large, potentially complicated scheduling problems. In this paper we consider a specific instance of such a problem: planning efficient utilization of the facilities and technicians that maintain the United States nuclear stockpile. In particular, Sandia National Laboratories has developed and implemented the Pantex Process Model (PPM) [7] to support planning activities at Pantex, a US Department of Energy (DOE) production plant in Amarillo, Texas. The plant simultaneously supports three major DOE programs – nuclear weapon disposal, stockpile evaluation, and stockpile maintenance – which share its facilities, technicians, and equipment. We focus on one piece of the PPM, namely the Evaluation Planning Module (EPM) which projects facility and technician utilization over a given planning horizon (typically a year).

A substantial portion of the Pantex workload relates to evaluation of weapons in the active stockpile. Each weapon evaluation involves partial disassembly of the weapon, one or more inspection tests, and then re-assembly and return of the weapon to the active stockpile.

A detailed study of this problem yields a complicated mixed-integer programming (MIP) model with more than hundreds of thousands of variables and even more constraints. (See Section 2 for a detailed problem description.) Pantex planners want (approximate) answers to planning queries in minutes, with the code running on a local PC. Consistently and quickly solving such a model exactly is impossible using today’s algorithms on a PC, and thus, in addition to using branch-and-bound, we must use good heuristics and approximation algorithms.

In this paper, we empirically evaluate the relative success of several approaches to solving the MIP. We employ traditional branch-and-bound algorithms to exactly solve the MIP. Also, we employ several different techniques in combination to approximately solve this mixed-integer program. These techniques all follow the same general structure:

1. Solve the LP relaxation of the MIP.
2. Use an α -point method to generate an ordering of the jobs.
3. Convert the ordering into a feasible schedule.
4. Try to improve the schedule via local improvements.

Particularly noteworthy is our use of α -points in this context (Step 2). Given a solution to a linear program, in which the variables x_{jt} denote the fraction of job j completed (or started) at time t , the α -point of job j is the earliest time at which an α fraction of job j has been completed (or started). Each value of α defines an ordering of the jobs; given one LP solution, it is possible

to generate many different orderings. α -points have been used extensively in designing approximation algorithms for combinatorial scheduling problems (see, e.g. [6, 4, 14, 8, 5]), and are explained in more detail in Section 3. Computational work by Savelsbergh, Uma and Wein has shown their practical value in settings where the objective is related to flow time or completion time [13]. α -points have also performed well in practice for the best-effort objective, where one finishes as many jobs as possible by their deadlines [12]. Möhring, Schulz, Stork, and Uetz use, in addition to other approximation techniques, α -points to design heuristics for problems with the makespan objective in a setting where jobs also need simultaneous processing [10] from different machines. This feature is the same in the EPM, and is called resource-constrained project scheduling. For a survey of resource-constrained project scheduling, see [1]. To our knowledge, there is no prior work on using α -points in problems with hard *deadlines*. These problems pose a particular challenge, as a greedy schedule created from a particular job ordering may not satisfy all the deadlines if it must obey precedence and resource constraints. Therefore, the model used by EPM allows (but seeks to minimize) resource overage. Because the EPM plans for future operations, this is not an artificial analysis tool; the goal is to acquire the resources by the time they are needed. Under this objective, we demonstrate the utility of α -points in a setting with deadlines and precedence constraints.

In a simple combinatorial scheduling problem, once we compute an ordering for the jobs, the task (Step 3) of computing a schedule is typically no more difficult than list-scheduling or some equally simple procedure. In this problem, the ordering is not sufficient to indicate a schedule, so we need to use a more sophisticated job assignment algorithm. We employ two basic approaches for job assignment. One is a greedy method while the other is a more sophisticated approach based on assignment and maximum flow subproblems. By contrasting these two approaches, we are able to evaluate the relative efficacy of using multiple α -points versus different levels of sophistication of job-assignment. We also employ a local search at the end of the algorithm.

We solve the MIPs within this paper using both the CPLEX 9.0 commercial MIP solver and the parallel Integer and Combinatorial Optimizer (PICO) which is under development at Sandia National Laboratories and RUTCOR [3]. PICO is a general branch-and-bound framework specialized to mixed-integer-programming with many computers in parallel as opposed to the single (or serial) computer version of CPLEX. We describe these issues in more detail in Section 3.

We design a suite of sample data and test the algorithms. We report on our data, experiments and conclusions in Sections 4, 5, and 6.

The goals of this project were twofold. First, we wanted to develop a program that could efficiently and accurately help with the Pantex planning problem. Second, within this challenging context, we wanted to experimentally test various ideas that were originally designed for “cleaner” problems. The details of this paper focus on the second goal, and we report on our particular conclusions in Section 6. In summary, we demonstrate the value of using α -points as a way

to quickly and cheaply generate many feasible solutions to an integer program from one solution of an LP relaxation. In this particular environment, the use of α -points, combined with other heuristics outperforms local search. We also see the value of finding combinatorially-structured subproblems as opposed to using simple greedy approaches.

2 Problem Formulation

In the Pantex planning problem, each job (evaluation) consists of a *set of tasks*, related by precedence constraints. The precedence graph is an *out-forest*. In practice the trees of precedence constraints are highly chainlike. Each task j has a fixed processing time p_j , which can be as short as an hour or as long as several months. It also has a release date r_j and a hard deadline d_j .

The time horizon is broken into weeks, currently consisting of six consecutive working days. Each day is eight hours long. Task lengths are given in hours.

Each task j must be performed without interruption in a facility of type k_j . While task j is being executed, a technician crew of size s_j must be assigned to that task. The actual technicians may change over the life of the task, but the crew size must always be maintained. Each technician in the crew must have the proper training to perform the task. Training is controlled by certifications. Each technician has a list of certifications to which (s)he can be assigned and each task j has a required certification c_j . Facility availability is given in hours per week and we assume all technicians are available at all times.

In a true schedule, each task is assigned to a specific facility and given a specific team of qualified technicians. However, for planning future technician/facility needs, it is currently sufficient to assign tasks to a pool of facilities and technicians checked at a weekly granularity. That is, a plan is feasible with respect to facility resources for a particular week if the total work for each facility type k is no more than the number of type- k facility hours available that week. This is equivalent to allowing preemption and migration. Using ideas similar to those used by McNaughton [9], we can show that this underestimates true facility requirements by at most a factor of two, as the preemptive schedule can be converted into a nonpreemptive schedule by taking the one job per machine that is preempted and placing it on a new machine. Technician hours are pooled similarly. To determine certification-hour availability, each technician is assigned to certifications by specifying the amount of time (possibly fractional) (s)he will devote to each certification during each week. No technician is assigned more hours for a particular certification during a week than the total lengths (within that week) of tasks requiring that certification.

We make the further restriction, consistent with Pantex's operations, that short tasks (less than a day) can start at any time of the day, but they cannot be split across two days; they must be completed on the day on which they start.

A *production plan* or schedule assigns a start time to each task. Since preemption is not allowed, a task occupies a facility for its entire duration beginning at its start time. A production plan is feasible if: a) all precedence constraints,

release dates, and deadlines are obeyed, b) short tasks are completely scheduled within a single day, c) the total amount of work scheduled for each facility type during any particular week does not exceed the availability of such facilities, d) for each week the requirements for technicians are matched by qualified technician assignments and each technician is assigned no more than a week of work, and e) for each week, no technician is assigned to a particular certification for more hours than the sum of the task lengths (within that week) of tasks requiring that certification. This last constraint partially prevents serialization of parallel task work, that is, allowing one worker to do the work of two provided he works twice as long.

A typical planning problem spans a year and involves at least 500 jobs and 1000 tasks. Each job has from one to six tasks. There are about 28 – 32 facility types and 300 technicians, each of whom holds 2–5 of the 80 – 100 possible certifications.

In practice, these planning problems are often infeasible. Future work can be performed by newly-hired or retrained technicians in facilities that are newly built or borrowed from other programs sharing Pantex’s facilities. Thus our objective is to find a feasible plan that minimizes the use of extra resources beyond what is currently available. Specifically, we wish to minimize the total number of hours assigned to *ghost facilities (facility overage)* and *ghost certification hours (technician overage)*. In general these terms could be weighted based upon the difficulty of acquiring facilities relative to technicians. However for this study, we weight the two overages equally.

The production plan is not a true schedule because we do not assign workers to specific tasks. We only guarantee that the pool of workers is sufficient in the aggregate. Given the uncertainty in the data (job load, task duration, etc), extra resolution is probably not warranted. However, the optimal overage for this schedule is a lower bound on the optimal overage for a true schedule where tasks are assigned to specific facilities with specific teams of technicians and all resources constraints are met at the finest resolution (in this case, by the hour). Therefore, a decision to allocate more resources to Pantex can be justified based on the optimal plan, but the resources specified in the plan may not be sufficient.

2.1 MIP Formulation of the Pantex Problem

Input Parameters/Constants/Shorthand

p_j, r_j, d_j	The processing time (in hours), release day and deadline day of task j
ρ_j	The minimum number of full days needed to process task j (i.e. $\rho_j = \lceil p_j/8 \rceil$).
$T(j)$	Set of possible start days for task j .
$p(j, t, \tau)$	Number of hours work on task j performed during week τ if j is started on day t .
$T(j, \tau)$	Set of start times t for task j such that $p(j, t, \tau) > 0$.
k_j, c_j, s_j	Facility type, technician certification and crew size for task j .
$C(w)$	The set of certifications held by worker w
$f_{k, \tau}$	The number of hours available in facilities of type k during week τ .
$b(j, j')$	1 if $j \prec j'$ and task j' can be “packed” with j , 0 otherwise

Variables

We use four types of variables. The x variables are integral, while the remaining variables are rational.

$x_{jt} = 1$ if task j starts on day t , and 0 otherwise

$y_{wct} =$ fraction of time worker w spends using certification c in week τ

$F_{k\tau} =$ number of ghost facilities of type k in week τ

$G_{c\tau} =$ number of ghost technicians with certification c in week τ

MIP Formulation

$$(TILP) \text{ minimize } \sum_{k,\tau} F_{k\tau} + \sum_{c,\tau} G_{c,\tau}$$

subject to

$$\sum_{t \in T(j)} x_{jt} = 1 \quad \forall j \quad (1)$$

$$\sum_{c \in C(w)} y_{wct} \leq 1 \quad \forall w, \tau \quad (2)$$

$$x_{jt} \leq \sum_{r_{j'} \leq t' \leq t - \rho_{j'} + b(j', j)} x_{j't'} \quad \forall t \in T(j), j' \prec j \quad (3)$$

$$\sum_{j:k=k_j} \sum_{t \in T(j,\tau)} p(j,t,\tau) x_{jt} \leq f_{k,\tau} + 48F_{k\tau} \quad \forall \tau, k \quad (4)$$

$$\sum_{j:c_j=c} \sum_{t \in T(j,\tau)} p(j,t,\tau) s_{jt} x_{jt} \leq \sum_w 48y_{wct} + 48G_{c\tau} \quad \forall \tau, c \quad (5)$$

$$48y_{wct} \leq \sum_{j:c_j=c} \sum_{t \in T(j,\tau)} p(j,t,\tau) x_{jt} \quad \forall c, \tau, w : c \in C(w) \quad (6)$$

$$x \in \{0, 1\} \quad (7)$$

Constraints (1) ensure that every task is done. Constraints (2) prevent over-scheduling any technician in any week. Constraints (3) ensure a task is not started until all predecessors are completed. Constraints (4) ensure there is sufficient facility capacity in each week to perform all the work that must be done in that week, and constraints (5) are the analogous constraints on certification hours. Constraints (6) prevent some situations where a technician is taking the place of multiple technicians. There are 48 work hours in a week. Although short jobs may start at any hour, as long as they fit in a day, we will always start them at the beginning of the day or immediately following the completion of their last predecessor (whichever is later). We will refer to this program as TILP.

3 Algorithms

In this section, we describe our algorithms. We also describe the branch and bound used to solve the mixed integer program TILP, which provided the optimal

objective values (or lower bounds) that we use as a basis for comparison for our heuristics. Our heuristics all followed the same general steps:

1. Solve the linear programming relaxation of TILP to obtain x_{jt}^* .
2. Call the BEST- α or FIXED- α subroutine to convert the TILP solution x_{jt}^* to a set, Π , of priority orderings for the jobs
3. For each ordering $\pi \in \Pi$, call the COMB or GREEDY subroutine to determine a schedule $\omega(\pi)$ for the tasks
4. Use local search to try to improve the objective for the schedules.

Steps 2 and 3 each employ two different subroutines resulting in four different algorithms. We implemented and tested each algorithm, and also examined the value of using local search to improve the solution.

Mixed-integer programming. To reduce the size of the MIP, task starts are represented at daily granularity rather than to the hour (the smallest size of a task). We force long jobs to start at the beginning of a day. This does not significantly decrease planning flexibility. However, if we were to force short jobs to align with the start of days, we could introduce significant unforced idle time, especially on chains of short jobs. Therefore, short jobs can start at hourly increments within a day. This is not explicitly represented in the MIP. Start times are interpreted as mid-day (earliest possible) whenever one or more predecessors are also running that day.

To enforce this interpretation, we must take some care in formulating precedence constraints. In the MIP given in Section 2, we enforce precedence $j \prec j'$ in constraints (3). Normally, we need only enforce a precedence constraint between immediate (predecessor,successor) pairs. However, when j and j' can finish on the same day (they pack, $b(j, j') = 1$), we must usually add an extra precedence constraint between j' and a more distant predecessor to guarantee j' is always scheduled completely within a day.

We used two different MIP solvers, CPLEX 9.0 and PICO, to solve the mixed-integer programs. CPLEX (serial) is the standard commercial MIP solver, whereas PICO (Parallel Integer Combinatorial Optimizer) is a massively-parallel mixed-integer programming solver being developed at Sandia National Laboratories and RUTCOR. PICO requires a linear-programming solver to compute lower bounds, for which we currently use the free solver, COIN LP. PICO is licensed under the GNU general library public license and is expected to be released free in 2004. We used only the serial version of PICO in this paper.

Neither solver could handle problems spanning the entire year-long time horizon, and so we tested our algorithms on problems spanning roughly half the time horizon (i.e. around six months). For these problems, both codes could solve most, but not all of the instances considered, and CPLEX was faster than PICO. Sections 4 and 5 contain more details about the size of problem versus tractability.

We also started developing routines specific to the MIP for use within PICO. For a fractionally assigned task j , we can branch on the full set of x_{jt} variables

where one child forces j to start by time t^* and the other child forces j to start after $t^* + 1$ where t^* is the α -point for $\alpha = 0.5$. One could get similar behavior with CPLEX by explicitly listing all decision variables for each job as a special-ordered set. We make branch choices based on gradients just as it would for branching on single variables. As described in Section 3, we can use an α -point-based algorithm to find feasible solutions early. This allows early pruning of the search tree.

In future work, we will use the parallel version of PICO in order to solve the full version of these problems, and test the ideas that we have implemented within PICO.

α -points. The IP code described in the previous section is not able to efficiently solve all of our instances. We therefore begin with the solution x_{jt}^* to the linear programming relaxation. In step 2, BEST- α and FIXED- α convert a solution x_{jt}^* to a set of orderings, Π , of the tasks. They both use the method of α -points to do this. The variables x_{jt} were defined in the formulation of TILP, and are an indicator of which day a given task starts in. After relaxing the integrality, the start-time variables x_{jt} now represent the fraction of the task j that starts at day t . Start-time variables can be converted into the more standard completion-time variables by the formula: $\bar{x}_{j(t+\rho_j)} = x_{jt}, \forall j, t$. Then, for a given $\alpha \in [0, 1]$, and solution to the linear programming relaxation of (TILP), the α -point of a task, \bar{t}_j^α , is defined as the first day t at which the fraction of task j completing at or before t exceeds α . Formally, this is:

$$\bar{t}_j^\alpha = \min\{t : \sum_{\tau \leq t} \bar{x}_{j\tau} \geq \alpha\}. \quad (8)$$

Alternatively, we can use the original time-indexed variables in equation (8):

$$t_j^\alpha = \min\{t : \sum_{\tau \leq t} x_{j\tau} \geq \alpha\}. \quad (9)$$

Thus, t_j^α represents the first day t at which the fraction of task j starting at or before t exceeds α . The task ordering is simply the (ascending) order of either t_j^α or \bar{t}_j^α . In general, using the orderings generated by \bar{t}_j^α will favor starting longer tasks later than shorter tasks. Our experiments showed that t_j^α resulted in a better objective than \bar{t}_j^α . This may be a result of the presence of due dates and the resource optimizing objective, which would favor starting longer tasks earlier.

Implementing these methods involves calculating t_j^α or \bar{t}_j^α from equations (9) or (8) for each α in some given set \mathcal{A} . The distinction between different α -points methods involve what \mathcal{A} is. Our heuristics used methods known as BEST- α and FIXED- α (see e.g. [13]). BEST- α corresponds to trying all $\alpha \in [0, 1]$ that yield different orderings. There are a finite number of α to try, as the orderings change at most once for every nonzero x_{jt} . For a basic solution to MILP, the number of nonzero x_{jt} is bound by the number of basic variables in the linear programming relaxation. For our particular implementations, we approximated BEST- α by using 51 α spread uniformly over the $[0, 1]$ interval. The FIXED- α method corresponds to setting \mathcal{A} to just one particular α . We set $\mathcal{A} = \{0.5\}$.

COMB. This section describes an algorithm, COMB, that converts an ordering, π , of the tasks to a feasible schedule, or plan, $\omega(\pi)$. In addition to the ordering, COMB uses information from the linear program relaxation and combinatorial structure to create schedules. This algorithm is the one used to generate early feasible solutions to strengthen pruning in (PICO-based) branch and bound.

Given an ordering π , the heuristic must first deal with a subtlety concerning the discretization of the problem. Specifically, for a short task j' that packs with its predecessor j , the starting time alpha point $t_{j'}^\alpha$ (in this case equivalent to $\bar{t}_{j'}^\alpha$) might be the same day in which task j finishes. Since the alpha points are expressed in days, not hours, task j completes sometime in the middle of day t_j^α when started at the beginning of day t_j^α . The true starting hour of task j' must be adjusted accordingly.

COMB computes the “resource availability” from the linear programming solution. Ghost facilities used in the LP solution are added to the true facility availability. Technician-hour availability for each certification for each week is the number of hours the LP assigns to that certification over all technicians for that week plus the ghost technician hours. Since the objective-function value of this solution is a lower bound on the optimal integer solution, we are free to use all these resources in our heuristic solution without degrading the objective value. Therefore, only resources used in excess of this availability are considered “overage” for this heuristic.

COMB schedules tasks in alpha-point order. To schedule a task j , COMB computes the incremental decrease in technician and facility availability incurred by each potential starting day of j and places j in the earliest place that creates no new overage. If no such place exists, COMB places the task either as early as possible, or in the place with minimum overage. The strategy is determined at the start of the heuristic; for this paper we try both and take the best result. COMB decrements the technician and facility availability for each week in the lifetime of the task and continues to the next task.

Given the greedily-determined start times, COMB computes an optimal (rational) technician assignment using one maximum-flow computation for each week. The flow problem is formulated as follows. There is a source node s , a node W_i for each worker i , a node C_j for each certification j and a sink t . The source is connected to each of the W_i nodes with capacity equal to the total time the worker is available to work during this week. In our current model, this is the same for all workers: 48 hours. There is an edge from each worker node W_i to each certification node C_j where worker i has certification j . The capacity is the total number of hours of work for certification j scheduled in this week (according to our heuristic schedule). Finally, there is an edge from each certification node to the sink with capacity equal to the total man-hours of work for certification j in this week. That is, for each (piece of a) task with certification j run in this time week, we multiply the length of the task (in this week) by the crew size. The capacity on the source-to-worker edges reflects the bound on technician availability. The capacity of the worker-to-certification edges reflects the total time a worker can spend on a certification (constraints 6 in MILP). The

capacity of the certification-to-sink edges reflects the total work requirement for each certification. The technician assignment is encoded in the flow from worker nodes to certification nodes. The residual capacity on the edge from certification j to the sink (that is the difference between the capacity and the flow) is the technician overage from this assignment. In particular, if the maximum flow in this network saturates all the certification-to-sink edges, then all the work is done and there is no overage.

Assigning resources greedily. The greedy subroutine for assigning resources, GREEDY, creates a feasible schedule, $\omega(\pi)$, from a given ordering, π . Note that $\omega(\pi)$ corresponds to a start time for each task, a schedule for each technicians and facility, and a number of ghost hours needed for each certification and facility per week.

We either assign all technicians to tasks first and then all facilities to tasks, or vice versa. Our experiments demonstrated the former is better. Because of these results, and the two methods' similarity, we only describe the TechSchedFirst subroutine. Due to space constraints we do not prove the algorithm's correctness.

Throughout TechSchedFirst, a set of jobs, \mathcal{J} , which are not scheduled and have no uncompleted predecessors are maintained. Also, each job's earliest possible start time is maintained. This is the minimum of its release date and the completion time of its predecessors. At each iteration, TechSchedFirst greedily assigns technicians to the job in \mathcal{J} with the highest priority according to π by determining the time in the time window of its earliest possible start time and due date that corresponds to the point at which the most technicians will be free. This is possible to do via interval trees (see Cormen et. al. 2001, [2]). If more technicians are required, the necessary amount of ghost technician hours are allocated. TechSchedFirst then updates \mathcal{J} and the earliest possible start times of all jobs. With the start times of each job established, TechSchedFirst then schedules facilities to each task in ascending start time order, with ghost facility hours scheduled as needed.

Local search. Since our experiments indicated that technicians were a more constrained resource, we implemented a straightforward local search heuristic, called LOCALIMPROVE, that swaps technicians between different tasks. We did not implement local search on COMB as our goal was to compare the more sophisticated job-assignment heuristic to greedy methods.

LOCALIMPROVE takes a feasible schedule as input and returns a feasible schedule with an objective no worse than the input. LOCALIMPROVE assumes tasks are sorted in descending ghost technician hours. The subroutine examines each task j , and, if it has ghost technicians, determines any other tasks that have scheduled process time overlap. It calculates the set, \mathcal{C}^C , of all tasks that have no overlap with j . Then $\mathcal{C} = \mathcal{J} \setminus \mathcal{C}^C$ is the set of tasks with overlap with j . Swapping technicians to task j only affects tasks in \mathcal{C} . It finds all technicians that are not currently working on j , but have the correct certification to work on j . It swaps the technician to j that corresponds to the best change in the objective, updating

the ghost requirements of both j and any task in \mathcal{C} on which the technician was working. This is iterated as long as j still has ghost technicians.

We also applied a simple evolutionary algorithm which did not work as well. In future work, we will consider more sophisticated evolutionary algorithms.

4 Data

Due to security reasons, access to actual data was severely restricted. However, we know many properties and characteristics of the real datasets. We created a problem generator that is easily modified and creates problem instances that capture properties of the real data.

Series (jobs)	Tasks	Crew	Certs	Techs	C.per.T.	Fac.Types	Fac.Hours
SeriesA - (20)	212	1-6	100	300	3-5	30	100
SeriesA - (100)	1060	1-6	100	300	3-5	30	100
SeriesA - (200)	2120	1-6	100	300	3-5	30	100
SeriesB - (20)	212	1-6	100	150	3-5	6	10
SeriesB - (100)	1060	1-6	100	750	3-5	6	50
SeriesB - (200)	2120	1-6	100	1500	3-5	6	100
SeriesC - (20)	212	1-2	100	300	3-5	30	100
SeriesC - (100)	1060	1-2	100	300	3-5	30	100
SeriesC - (200)	2120	1-2	100	300	3-5	30	100

Table 1. Overview of selected problems

The problem generator uses predefined job templates that describe the successor-predecessor relationships of the tasks and include parameters for the size and requirements of each task. These templates were created using our knowledge that in the actual problem the jobs are highly chainlike, each task can have at most one predecessor and most tasks have only one successor even though we allow tasks to have more than one successor. We then combined this with templates for the facilities, technicians and certifications and the scenario, which defines the random distributions of the task sizes along with the time horizon. The actual size of the tasks is determined using a uniform random distribution, whose parameters are dependent on the scenario and the task. To select the certification and facility for a task, we split the certifications and facilities into a few sets and then randomly pick a certification and facility.

Due to hardware and time issues we created series of scaled down problems that retain most of the characteristics of the real problems. We needed to make the problems small enough so that the branch and bound could solve most of the instances, while keeping them large enough to contain all the original structure. The time horizon was cut down from a year to 6 months, and job lengths scaled down so that we fit the same number of jobs in this time horizon. To make the problems more challenging, we made the job templates larger and

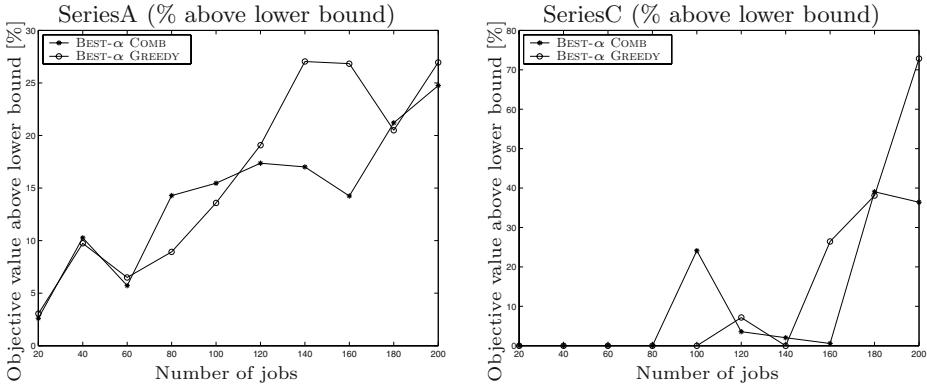


Fig. 1. Performance on SeriesA and SeriesC. We show the performance of the algorithms for SeriesA and SeriesC as a percentage of the objective value above the best known lower bound. The best known lower bound is the optimal IP solution for all problems except 160 – 200 jobs of SeriesA.

less chainlike than in the actual problem. We created three sets of problems, with 10 problems in each set. The goal of the first dataset, “Series A,” was to see how the algorithms would handle increasingly difficult problems that share the same fundamental properties. In some sense, the problem with 100 jobs is the most like the actual EPM problem. The next set of problems, “SeriesB”, was created to see how the algorithms would deal with problems of increasing size but not increasing difficulty. Since Series A and B seemed to require more technicians than facilities, we created “Series C” by decreasing the number of ghost technicians that the solutions requires and hence increased the importance of the ghost facilities. Table 1 shows a small, medium and large problem of each problem set. Finally, we also generated a full-year problem to test whether our heuristics would be able to solve a problem of this size. To create the full-year problem, we used the Series A problem with 100 jobs and doubled the time horizon and processing time of each job. We discuss the results of this small test in Section 5.

5 Results

We use results from the three data series described in Section 4 to illustrate the effectiveness of the algorithms, since the results from those series are representative of all the problems that we generated. Because of limited space, we focus mainly on the common conclusions, instead of analyzing in detail any specific results for the algorithms.

We ran GREEDY on a 1.8 GHz Pentium 4 with 512 MB of RAM running Linux kernel 2.4.7-10. The SPECint2000 and SPECint-base2000, performance comparisons are, respectively, 619 and 599. CPLEX 9.0, PICO, and COMB ran on a 3 GHz Xeon with 2 GB of RAM running Linux kernel 2.4.20-18. The

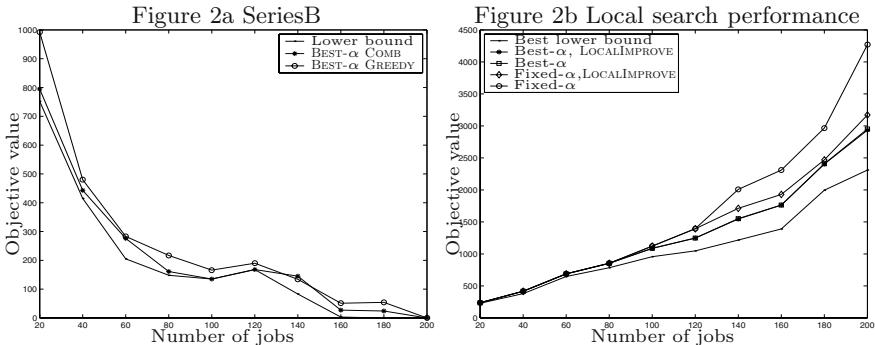


Fig. 2. Performance on SeriesB and the effect of local search. The graph on the left shows the results for SeriesB. The lower bound is the optimal IP solution for 20 – 140 jobs problems. Since the lower bound is zero for the 160 – 200 jobs problems, we show the actual objective values for all the algorithms and the lower bound. The graph on the right shows The effect of LOCALIMPROVE on BEST- α and FIXED- α , using GREEDY.

SPECint2000 and SPECint-base2000 performance comparisons are, respectively, 1138 and 1089. Because of the considerable difference in speed between the two machines, we only analyzed runtimes for CPLEX 9.0, PICO, and COMB. More information on the SPEC performance comparison benchmarks can be found at <http://www.specbench.org>.

Figure 1 shows the performance of the heuristics BEST- α and GREEDY as a percentage of the objective value above the best known lower bound. In most cases this lower bound is the optimal integer solution. However the most difficult problems (SeriesA with 160,180 and 200 jobs, SeriesB with 160,180 and 200 jobs) were too big to be solved optimally in a reasonable amount of time and for these we give the best lower bound encountered during the branch-and-bound computation. Figure 2a shows the performance of BEST- α and GREEDY along with the best known lower bound for SeriesB. Since the last problems of SeriesB have lower bound of 0, we cannot find the percentage over the best known lower bound for these problems. Figure 2b shows the effect of using LOCALIMPROVE with the heuristics.

In Series A, CPLEX required seconds for the 20, 40, and 60 job problems, minutes for the 80 and 100 job problems hours for the 120 and 140 job problems and could not solve the 160, 180, and 200 job problems in under 48 hours. In Series B, CPLEX required seconds for the 20, 40, 60, 80, and 100 job problems, minutes for the 120 and 140 job problems, and did not solve the 160, 180, and 200 job problems in under 48 hours. CPLEX solved the 20 to 160 job problems of Series C in at most 72 seconds and required several minutes for the 180 and 200 job problems. CPLEX's success with the Series C problem was related to the fact that it did not require any branches in the branch and bound tree - the optimal solution to the linear program was integral. The heuristics ran on

all problems in Series A in under 133 seconds, in Series B in under 321 seconds, and under 130 seconds for all problems in Series C.

BEST- α is better than FIXED- α . In almost every instance where both heuristics do not achieve the optimal solution, BEST- α yields significant improvement over FIXED- α . Figure 2b shows this for SeriesA; the results for other series were similar. The difference increases as the problems become more difficult. In the extreme case the solution from FIXED- α without LOCALIMPROVE is more than factor of 4 times the solution from BEST- α without LOCALIMPROVE.

COMB performs better than GREEDY. For SeriesA, COMB was on average 14.3% over the optimal solution, compared to 16.2% for GREEDY. For SeriesC, the average was 10.5% for COMB and 14.5% for GREEDY. Of the 10 instances in SeriesB, COMB performed better in 8 instances, while GREEDY only performed better in only one instance. This can been seen in Figures 1 and 2a.

LOCALIMPROVE helps FIXED- α but it does not help BEST- α . Figure 2b shows the effect of LOCALIMPROVE for SeriesA. The results for other series were similar. For FIXED- α , LOCALIMPROVE has a significant improvement on the solution, while the effect on BEST- α is minimal. Of the 30 instances, only 3 instances had improved solutions when using BEST- α with LOCALIMPROVE compared to using only BEST- α . Also while LOCALIMPROVE significantly improves the solution of FIXED- α , there was not a single instance where FIXED- α with LOCALIMPROVE found a better solution than BEST- α without LOCALIMPROVE.

A small decrease in accuracy gives a huge decrease in running time. CPLEX required several hours to solve the most challenging tractable problems in these series. Moreover, the most difficult problems ran for more than two days without a solution. The heuristics scale efficiently with the problem input size and run in minutes at worst; we get a solution to the most difficult problem in under 7 minutes. In addition, the heuristics usually return a solution that is within 10% of the optimal value.

The best heuristic is BEST- α COMB. Overall, our results indicate that BEST- α and COMB is the best combination. LOCALIMPROVE does not yield any significant improvements on the BEST- α and its running time hardly justifies its inclusion. Hence the best combination in terms of speed and results is BEST- α COMB.

We tested BEST- α GREEDY on our full-year problem and found that it computed a solution within 14% of the linear programming relaxation lower-bound. Based on the results of the the other experiments, we hypothesize that BEST- α COMB would perform well on the larger problems as well.

6 Conclusion and Future Work

We have provided evidence that sophisticated algorithmic ideas, designed for combinatorial scheduling problems and designed with approximation algorithms in mind, can be used in heuristics to obtain good solutions to a difficult, “real-life” project scheduling problem. In addition to providing solutions in a fraction of the time of exact methods, our heuristics allow a finer granularity of time.

Our experiments also provide evidence that α -points can yield reasonable approximations for a difficult project scheduling problem. This supports the idea that α -points can be practically used in several problem contexts. Our experiments demonstrate that, for this problem, BEST- α is a significant improvement on FIXED- α , and multiple choices for α should be used in any algorithm employing α -points. In particular, α -points provide an inexpensive way to generate many feasible schedules from one linear program. We have also demonstrated that α -points can be used in situations with hard deadlines.

In the future, we plan to run our heuristics and exact methods on full-scale data. We would like to compare the overage of an optimal plan to the overage for an optimal true schedule. We wish to quantify the effect of improved heuristics, and of rerunning heuristics on multiple search-tree nodes, on running times of exact methods. Additionally, we would like to consider different priority rules, especially some suggested for the resource leveling objective by Neumann and Zimmerman[11]. We are particularly interested in testing greatest resource demand or greatest resource demand per time unit. Finally, we would like to investigate the use of start-time based α -points further, both experimentally and theoretically.

References

- [1] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal on Operations Research*, 112:3–41, 1999.
- [2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press and McGraw-Hill, second edition, 2001.
- [3] Jonathan Eckstein, Cynthia A. Phillips, and William E. Hart. PICO: an object-oriented framework for parallel branch and bound. In *Inherently parallel algorithms in feasibility and optimization and their applications (Haifa, 2000)*, volume 8 of *Stud. Comput. Math.*, pages 219–265. North-Holland, Amsterdam, 2001.
- [4] M. Goemans. Improved approximation algorithms for scheduling with release dates. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 591–598, 1997.
- [5] Michel X. Goemans, Maurice Queyranne, Andreas S. Schulz, Martin Skutella, and Yaoguang Wang. Single machine scheduling with release dates. *SIAM J. Discrete Math.*, 15(2):165–192 (electronic), 2002.
- [6] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22:513–544, August 1997.
- [7] Edwin A. Kjeldgaard, Dean A. Jones, George F. List, Mark A. Turnquist, James W. Angelo, Richard D. Hopson, John Hudson, and Terry Holeman. Swords into plowshares: Nuclear weapon dismantlement, evaluation, and maintenance at pantex. *Interfaces*, 30(1):57–82, 2000.
- [8] F. Margot, M. Queyranne, and Y. Wang. Decompositions, network flows and a precdendce constrained single machine scheduling problem. talk by M. Queyranne at IMPS 97, 1997.

- [9] R. McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6:1–12, 1959.
- [10] Rolf H. Möhring, Andreas S. Schulz, Frederik Stork, and Marc Uetz. Resource-constrained project scheduling: computing lower bounds by solving minimum cut problems. In *Algorithms—ESA '99 (Prague)*, volume 1643 of *Lecture Notes in Comput. Sci.*, pages 139–150. Springer, Berlin, 1999.
- [11] Klaus Neumann and Jürgen Zimmermann. Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. *European Journal on Operations Research*, 127:425–443, 2000.
- [12] Cynthia A. Phillips, R. N. Uma, and Joel Wein. Off-line admission control for general scheduling problems. *Journal of Scheduling*, 3(6):365 – 382, 2000.
- [13] Martin W.P. Savelsbergh, R.N.Uma, and Joel Wein. An experimental study of LP-based scheduling heuristics. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 453–461, 1998.
- [14] A. S. Schulz and M. Skutella. Scheduling-LPs bear probabilities: Randomized approximations for min-sum criteria. In R. Burkard and G. Woeginger, editors, *Algorithms – ESA '97*, volume 1284 of *LNCS*, pages 416 – 429. Springer, Berlin, 1997. Proceedings of the 5th Annual European Symposium on Algorithms.

Three Min-Max Theorems Concerning Cyclic Orders of Strong Digraphs

Stéphane Bessy and Stéphan Thomassé

Laboratoire LaPCS, Université Claude Bernard Lyon 1,
50, avenue Tony Garnier, 69007 Lyon, France.
bessy@univ-lyon1.fr, thomasse@univ-lyon1.fr

Abstract. In 1963, Tibor Gallai [9] asked whether every strongly connected directed graph D is spanned by α directed circuits, where α is the stability of D . We give a proof of this conjecture using a new structure for digraphs called *cyclic orders*. Three min-max theorems for cyclic orders are derived.

1 A Conjecture of Gallai

In this paper, circuits of length two are allowed. Since loops and multiple arcs play no role in this topic, we will simply assume that our digraphs are loopless and simple. A directed graph (digraph) is *strongly connected*, or simply *strong*, if for all vertices x, y , there exists a directed path from x to y . A *stable set* of a directed graph D is a subset of vertices which are not pairwise joined by arcs. The *stability* of D , denoted by $\alpha(D)$, is the number of vertices of a maximum stable set of D . It is well-known, by the Gallai-Milgram theorem [10] (see also [1] p. 234 and [3] p. 44), that D admits a vertex-partition into $\alpha(D)$ disjoint paths. We shall use in our proof a particular case of this result, known as Dilworth's theorem [8]: a partial order P admits a vertex-partition into $\alpha(P)$ chains (linear orders). Here $\alpha(P)$ is the size of a maximal antichain. In [9], Gallai raised the problem, when D is strongly connected, of spanning D by a union of circuits. Precisely, he made the following conjecture (also formulated in [1] p. 330, [2] and [3] p. 45):

Conjecture 1 *Every strong digraph with stability α is spanned by the union of α circuits.*

The case $\alpha = 1$ is Camion's theorem [6]: Every strong tournament has a hamilton circuit. The case $\alpha = 2$ is a corollary of a result of Chen and Manalastas [7] (see also Bondy [4]): Every strong digraph with stability two is spanned by two circuits intersecting each other on a (possibly empty) path. In [11] was proved the case $\alpha = 3$. In the next section of this paper, we will give a proof of Gallai's conjecture for every α .

2 Cyclic Orders

An *enumeration* E of a digraph D on the vertex set V with $|V| = n$ is a bijection of V into the n -gon $\{1, \dots, n\}$. If $E(x) = k$ for a vertex x of D we simply denote x by v_k and write $E = v_1, \dots, v_n$. An arc $v_i v_j$ of D is a *forward arc* for E if $i < j$, otherwise it is a *backward arc* for E . A *forward path* for E is a directed path of D which only contains forward arcs.

Given a circuit C of the digraph D , the *index of C for E* is the number of turns of $E(C)$ (winding number), that is to say, the number of backward arcs for E contained in C . We denote this index by $i_E(C)$.

Clearly, cyclic permutations of v_1, \dots, v_n leave the index of every circuit of D unchanged. This is also the case when one permute two consecutive vertices of v_1, \dots, v_n which are not joined by an arc of D .

We say that two enumerations of D are *equivalent* if we can obtain one from the other by applying a sequence of the two previous operations. Finally, a *cyclic order of D* is an equivalence class of enumerations. For a circuit C of D , by construction, the index of C in any enumeration of a fixed cyclic order \mathcal{C} is the same and we denote it by $i_{\mathcal{C}}(C)$. A circuit of D with index 1 in \mathcal{C} is called *simple in \mathcal{C}* .

A cyclic order \mathcal{C} is *coherent* if every arc of D belongs to a simple circuit, or equivalently, if for every enumeration E of \mathcal{C} and every backward arc $v_j v_i$ for E , there exists a forward path for E from v_i to v_j . We denote by $\text{cir}(D)$ the set of all directed circuits of D , and for a set of circuits \mathcal{S} we simply denote by $i_{\mathcal{C}}(\mathcal{S})$ the sum of the index of the circuits of \mathcal{S} .

Lemma 1 *Every strong digraph has a coherent cyclic order.*

Proof. Let us consider a cyclic order \mathcal{C} which is minimum with respect to $i_{\mathcal{C}}(\text{cir}(D))$. We suppose for contradiction that \mathcal{C} is not coherent. There exists an enumeration $E = v_1, \dots, v_n$ and a backward arc $a = v_j v_i$ which is not in a simple circuit. Assume moreover that E and a are chosen in order to minimize $j - i$. Let k be the largest integer $i \leq k < j$ such that there exists a forward path from v_i to v_k . Observe that v_k has no out-neighbour in $]v_k, v_j]$. If $k \neq i$, by the minimality of $j - i$, v_k has no in-neighbour in $]v_k, v_j]$. In particular the enumeration $E' = v_1, \dots, v_{k-1}, v_{k+1}, \dots, v_j, v_k, v_{j+1}, \dots, v_n$ is equivalent to E , and contradicts the minimality of $j - i$. Thus $k = i$, and by the minimality of $j - i$, there is no in-neighbour of v_i in $]v_i, v_j]$. In particular the enumeration $E' = v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_{j-1}, v_i, v_j, \dots, v_n$ is equivalent to E . Observe now that in $E'' = v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_{j-1}, v_j, v_i, \dots, v_n$, every circuit C satisfies $i_{E''}(C) \leq i_{E'}(C)$, and the inequality is strict if the arc a belongs to C , a contradiction. \square

A direct corollary of Lemma 1 is that every strong tournament has a hamilton circuit, just consider for this any coherent cyclic order.

3 Three Min-Max Theorems

In this section, we will translate different invariants of graph theory in terms of cyclic order. From now on, \mathcal{C} will be a given coherent cyclic order of some strong digraph D .

The *cyclic stability* of \mathcal{C} is the maximum k for which there exists an enumeration v_1, \dots, v_n of \mathcal{C} such that $\{v_1, \dots, v_k\}$ is a stable set of D . We denote it by $\alpha(\mathcal{C})$, observe that we clearly have $\alpha(\mathcal{C}) \leq \alpha(D)$.

Lemma 2 *For a fixed enumeration of \mathcal{C} , let X be a subset of vertices of D such that there is no forward path for this enumeration between two distinct vertices of X . Then $|X| \leq \alpha(\mathcal{C})$.*

Proof. We consider an enumeration $E = v_1, \dots, v_n$ of \mathcal{C} such that there is no forward path between two distinct vertices of X , and chosen in such a way that $j - i$ is minimum, where v_i is the first element of X in the enumeration, and v_j is the last element of X in the enumeration. Suppose for contradiction that $X \neq \{v_i, \dots, v_j\}$. There exists $v_k \notin X$ for some $i < k < j$. There cannot exist both a forward path from $X \cap \{v_i, \dots, v_{k-1}\}$ to v_k and a forward path from v_k to $X \cap \{v_{k+1}, \dots, v_j\}$. Without loss of generality, we assume that there is no forward path from $X \cap \{v_i, \dots, v_{k-1}\}$ to v_k . Suppose moreover that v_k is chosen with minimum index k . Clearly, v_k has no in-neighbour in $\{v_i, \dots, v_{k-1}\}$, and since \mathcal{C} is coherent, v_k has no out-neighbour in $\{v_i, \dots, v_{k-1}\}$. Thus the enumeration $v_1, \dots, v_{i-1}, v_k, v_i, \dots, v_{k-1}, v_{k+1}, \dots, v_n$ belongs to \mathcal{C} , contradicting the minimality of $j - i$. Consequently, $X = \{v_i, \dots, v_j\}$, and there is no forward arcs, and then no backward arcs, between the vertices of X . Considering now the enumeration $v_i, \dots, v_n, v_1, \dots, v_{i-1}$, we conclude that $|X| \leq \alpha(\mathcal{C})$. \square

Let $P = x_1, \dots, x_k$ be a directed path, we call x_1 the *head* of P and x_k the *tail* of P . We denote the restriction of P to $\{x_i, \dots, x_j\}$ by $P[x_i, x_j]$.

The *min vertex cover by circuits* is the minimum index of a family of circuits which spans V . We denote it by $vspan(\mathcal{C})$.

Theorem 1 $\alpha_c(\mathcal{C}) = vspan(\mathcal{C})$.

Proof. Denote by k the cyclic stability of \mathcal{C} and let $E = v_1, \dots, v_n$ be an enumeration of \mathcal{C} such that $S = \{v_1, \dots, v_k\}$ is a stable set of D . Clearly, if a circuit C contains q vertices of S , the index of C is at least q . In particular the inequality $i_C(S) \geq k$ is satisfied for every spanning set of circuits of D and thus, $vspan(\mathcal{C}) \geq \alpha_c(\mathcal{C})$. To prove that equality holds, we will provide a spanning set \mathcal{S} of circuits of D with index $\alpha_c(\mathcal{C})$. Consider the auxiliary acyclic digraph D' on vertex set $V \cup \{v'_1, \dots, v'_k\}$ which arc set consists of every forward arc of E and every arc $v_i v'_j$ for which $v_i v_j$ is an arc of D . We call T' the transitive closure of D' . Let us prove that the size of a maximal antichain in the partial order T' is exactly k . Consider such an antichain A , and set $A_1 := A \cap \{v_1, \dots, v_k\}$, $A_2 := A \cap \{v_{k+1}, \dots, v_n\}$ and $A_3 := A \cap \{v'_1, \dots, v'_k\}$. Since one can arbitrarily permute

the vertices of S in the enumeration E and still remain in \mathcal{C} , we may assume that $A_3 = \{v'_1, \dots, v'_j\}$ for some $0 \leq j \leq k$. Since every vertex is in a simple circuit, there is a directed path in D' from v_i to v'_i , and consequently we cannot both have $v_i \in A$ and $v'_i \in A$. Clearly, the enumeration $E' = v_{j+1}, \dots, v_n, v_1, \dots, v_j$ belongs to \mathcal{C} . By the fact that A is an antichain of T' , there is no forward path joining two elements of $(A \cap V) \cup \{v_1, \dots, v_j\}$ in E' , and thus, by Lemma 2, $|A| = |(A \cap V) \cup \{v_1, \dots, v_j\}| \leq k$. Observe also that $\{v_1, \dots, v_k\}$ are the sources of T' and $\{v'_1, \dots, v'_k\}$ are the sinks of T' , and both are maximal antichains of T' . We apply Dilworth's theorem in order to partition T' into k chains (thus starting in the set $\{v_1, \dots, v_k\}$ and ending in the set $\{v'_1, \dots, v'_k\}$), and by this, there exists a spanning set P_1, \dots, P_k of directed paths of D' with heads in $\{v_1, \dots, v_k\}$ and tails in $\{v'_1, \dots, v'_k\}$. We can assume without loss of generality that the head of P_i is exactly v_i , for all $i = 1, \dots, k$. Let us now denote by σ the permutation of $\{1, \dots, k\}$ such that $v'_{\sigma(i)}$ is the tail of P_i , for all i . Assume that among all spanning sets of paths, we have chosen P_1, \dots, P_k (with respective heads v_1, \dots, v_k) in such a way that the permutation σ has a maximum number of cycles. We claim that if (i_1, \dots, i_p) is a cycle of σ (meaning that $\sigma(i_j) = i_{j+1}$ and $\sigma(i_p) = i_1$), then the paths P_{i_1}, \dots, P_{i_p} are pairwise vertex-disjoint. If not, suppose that v is a common vertex of P_{i_l} and P_{i_m} , and replace P_{i_l} by $P_{i_l}[v_{i_l}, v] \cup P_{i_m}[v, v_{\sigma(i_m)}]$ and P_{i_m} by $P_{i_m}[v_{i_m}, v] \cup P_{i_l}[v, v_{\sigma(i_l)}]$. This is a contradiction to the maximality of the number of cycles of σ . Now, in the set of paths P_1, \dots, P_k , contract all the pairs $\{v_i, v'_i\}$, for $i = 1, \dots, k$. This gives a spanning set \mathcal{S} of circuits of D which satisfies $i_{\mathcal{C}}(\mathcal{S}) = k$. \square

Corollary 11 *Every strong digraph D is spanned by $\alpha(D)$ circuits.*

Proof. By Lemma 1, D has a coherent cyclic order \mathcal{C} . By Theorem 1, D is spanned by a set \mathcal{S} of circuits such that $|\mathcal{S}| \leq i_{\mathcal{C}}(\mathcal{S}) = \alpha(\mathcal{C}) \leq \alpha(D)$. \square

We now establish the arc-cover analogue of Theorem 1. Again, a minimax result holds.

We denote by $\beta(\mathcal{C})$ the maximum k for which there exists an enumeration of \mathcal{C} with k backward arcs. We call k the *maximal feedback arc set* of \mathcal{C} . The *min arc cover by circuits* is the minimum index of a family of circuits which spans the arc set of D . We denote it by $aspan(\mathcal{C})$.

Corollary 12 $\beta(\mathcal{C}) = aspan(\mathcal{C})$.

Proof. Apply Theorem 1 to the line digraph of D . \square

The last min-max theorem consists of a fractional version of a theorem of J.A. Bondy ([5]). Our proof is similar to the classical proof on the circular chromatic number in the non-oriented case, see X. Zhu ([12]) for a survey.

The *cyclic chromatic number* of \mathcal{C} , denoted by $\chi(\mathcal{C})$, is the minimum k for which there exists an enumeration $E = v_1, \dots, v_{i_1}, v_{i_1+1}, \dots, v_{i_2}, v_{i_2+1}, \dots, v_{i_k}$ of \mathcal{C} for which $v_{i_j+1}, \dots, v_{i_{j+1}}$ is a stable set for all $j = 0, \dots, k - 1$ (with $i_0 := 0$).

Under the same hypothesis, the *circular chromatic number* of \mathcal{C} , denoted by $\chi_c(\mathcal{C})$ is the infimum of the numbers $r \geq 1$ for which \mathcal{C} admits an r -circular coloration. A mapping $f : V \rightarrow [0, r]$ is called an *r -circular coloration* if f verifies:

- 1) If x and y are linked in D , then $1 \leq |f(x) - f(y)| \leq r - 1$.
- 2) If $0 \leq f(v_1) \leq f(v_2) \leq \dots \leq f(v_n) < r$, then v_1, \dots, v_n must be an enumeration of \mathcal{C} . Such an enumeration is called *related to f* .

As usual, it is convenient to represent such an application as a mapping from V into a circle of the euclidean plane with circumference r . Condition 1) asserts then that two linked vertices have distance at least 1 on this circle. And by condition 2), the vertices of D are placed on the circle according to an enumeration of the cyclic order \mathcal{C} . By compactness of this representation, the infimum used in the definition of χ_c is a minimum, that is to say that there exists a $\chi_c(\mathcal{C})$ -circular coloration of \mathcal{C} .

Note that the enumeration given by 2) is possibly not unique. Indeed, two vertices of V may have the same image by f . In this case, these two vertices are not linked in D (because of 1)) and so, the two enumerations are equivalent. Moreover, two enumerations related to f have same sets of forward arcs and backward arcs.

Lemma 3 *For D a strong digraph and \mathcal{C} a coherent cyclic order of D , we have $[\chi_c(\mathcal{C})] = \chi(\mathcal{C})$.*

The following Lemma gives a criterion to decide whether an r -circular coloration f is best possible or not. We define an auxiliary digraph D_f with vertex set V and arc set $\{xy \in E(D) : f(y) - f(x) = 1 \text{ or } f(x) - f(y) = r - 1\}$. Observe that the arcs xy of D_f with $f(y) - f(x) = 1$ (resp. $f(x) - f(y) = r - 1$) are forward (resp. backward) in any enumeration related to f .

Lemma 4 *If f is an r -circular coloration of \mathcal{C} with $r > 2$ for which D_f is an acyclic digraph, then we can provide a real number $r' < r$ such that \mathcal{C} admits an r' -circular coloration.*

Proof. First of all, if a vertex x of D has an out-neighbour y with $f(x) - f(y) = 1$, by property 1) of f and coherence of \mathcal{C} , the arc yx must be also in D , and similarly if x has an out-neighbour y with $f(y) - f(x) = r - 1$, the arc yx must be in D . So, a vertex x with in-degree 0 (resp. out-degree 0) in D_f has no neighbour z with $f(x) - f(z) = 1$ modulo r (resp. $f(z) - f(x) = 1$ modulo r). Then, if $E(D_f) = \emptyset$, it is easy to provide an r' -circular coloration f' of \mathcal{C} with $r' < r$. Just multiply f by a factor $1 - \epsilon$ with $\epsilon > 0$ and ϵ small enough.

Now, amongst the r -circular colorations f of \mathcal{C} for which D_f is acyclic, choose one with minimal number of arcs for D_f . Assume that $E(D_f) \neq \emptyset$. We can choose a vertex x of D_f with in-degree 0 and out-degree at least 1. Denote by y_1, \dots, y_p the out-neighbours of x in D_f , we have for all i , $f(y_i) = f(x) + 1$ modulo r . By definition of f , x has no neighbour z such that $f(x) - f(z) < 1$ or

$f(z) - f(x) > r - 1$ and, moreover, since x has in-degree 0 in D_f , by the previous remark, x has no neighbour z with $f(x) - f(z) = 1$ or $f(z) - f(x) = r - 1$. Observe that none of the y_i verifies this, because $r > 2$. So, we can provide an r -circular coloration f' derived from f just by changing the value of $f(x)$: choose $f'(x) = f(x) - \epsilon$ modulo r with $\epsilon > 0$ and such that no neighbour of x has an image by f in $[f'(x) - 1, f'(x)] \cup [r - 1 + f'(x), r]$. We check that $E(D_{f'}) = E(D_f) \setminus \{xy_i : i = 1, \dots, p\}$, which contradicts the choice of f . So, $E(D_f) = \emptyset$ and we provide an r' -circular coloration of \mathcal{C} with $r' < r$ as previously. \square

Finally, we can state a third min-max theorem about cyclic orders. For this, we define, for a fixed cyclic order \mathcal{C} , the *cyclic length* of a circuit C of D , denoted by $lc(C)$, as the number of vertices of C in D , divided by the index of C in \mathcal{C} . The *max circuit length* in \mathcal{C} is the maximum $lc(C)$ for a circuit C of D . We denote it by $l_c(\mathcal{C})$.

Theorem 2 $\chi_c(\mathcal{C}) = l_c(\mathcal{C})$.

Proof. Consider an r -circular coloration f of \mathcal{C} with $r = \chi_c(\mathcal{C})$ and $E = v_1, \dots, v_n$ an enumeration of \mathcal{C} related to f . For a circuit C in D , we compute the length l of the image of C by f :

$$l := \sum_{\substack{xy \in E(C) \\ xy \text{ forward in } E}} (f(y) - f(x)) + \sum_{\substack{xy \in E(C) \\ xy \text{ backward in } E}} (r + f(y) - f(x))$$

A straightforward simplification of the sum gives $l = r \cdot ic(C)$. Furthermore, condition 1) of the definition of f implies that $f(y) - f(x) \geq 1$ if $xy \in E(C)$ and xy is forward in E (i.e. $f(x) < f(y)$) and $r + f(y) - f(x) \geq 1$ if $xy \in E(C)$ and xy is backward in E (i.e. $f(y) < f(x)$). So, we have $l \geq r \cdot ic(C) \geq l(C)$, hence $r \geq lc(C)$, and the inequality $\chi_c(\mathcal{C}) \geq \max\{lc(C) : C \text{ circuit of } D\}$ holds.

To get the equality, we have to find a circuit C of D such that $lc(C) = r$. First of all, since \mathcal{C} is coherent, it has a circuit of index 1 and then, its cyclic length is greater or equal to 2. Thus, the previous inequality gives $r \geq 2$ and so, states the case $r = 2$. From now on, assume that $r > 2$, Lemma 4 asserts that there exists a circuit C in the digraph D_f . So, the inequalities provided in the direct sens of the proof are equalities: the image of every arc of C by f has length 1 if the arc is forward or $r - 1$ if the arc is backward. So, we have $l = l(C)$, and $lc(C) = l(C)/ic(C) = r$, which achieves the bound. \square

A corollary of Theorem 2 is an earlier result of J.A. Bondy, known since 1976. The *chromatic number* of a digraph D , denoted by $\chi(D)$, is the minimal number k such that the vertices of D admit a partition into k stable sets of D . Clearly, $\chi(D) \leq \chi_c(\mathcal{C})$.

Corollary 21 (Bondy [5]) *Every strong digraph D has a circuit with length at least $\chi(D)$.*

Proof. Consider a coherent cyclic order \mathcal{C} for D and apply Theorem 2 to provide a circuit C with $l_C(C) = \chi_c(\mathcal{C})$. Since by Lemma 3 $\lceil l_C(C) \rceil = \lceil \chi_c(\mathcal{C}) \rceil = \chi(\mathcal{C})$, we get $\chi(D) \leq \chi(\mathcal{C}) = \lceil l_C(C) \rceil \leq l(C)$. \square

We gratefully thank J.A. Bondy who told us that a link could exist between [5] and Gallai's problem.

A full version of the paper will appear in Combinatorica.

References

1. J. Bang-Jensen and G. Gutin, Digraphs. Theory, algorithms and applications, *Springer Monographs in Mathematics*, Springer-Verlag London, 2001.
2. C. Berge, Path partitions in directed graphs, *Ann. Discrete Math.*, **17** (1983), 59–63.
3. J.A. Bondy, Basic graph theory: paths and circuits. *Handbook of combinatorics*, Vol. 1, 2, Elsevier, Amsterdam, 1995.
4. J.A. Bondy, A short proof of the Chen-Manalastas theorem, *Discrete Maths*, **146** (1995), 289–292.
5. J.A. Bondy, Diconnected orientations and a conjecture of Las Vergnas, *J. London Math. Soc. (2)*, **14** (1976), 277–282.
6. P. Camion, Chemins et circuits hamiltoniens des graphes complets, *C. R. Acad. Sci.*, **249** (1959), 2151–2152.
7. C.C. Chen and P. Manalastas, Every finite strongly connected digraph of stability 2 has a Hamiltonian path, *Discrete Mathematics*, **44** (1983), 243–250.
8. R.P. Dilworth, A decomposition theorem for partially ordered sets, *Annals of Mathematics*, **51** (1950), 161–166.
9. T. Gallai, Problem 15, in: M. Fiedler, ed., *Theory of Graphs and its Applications* (Czech. Acad. Sci. Prague, 1964), 161.
10. T. Gallai and A.N. Milgram, Verallgemeinerung eines graphentheoretischen Satzes von Rédei, *Acta Sci. Math. Szeged*, **21** (1960), 181–186.
11. S. Thomassé, Spanning a strong digraph by α cycles, the case $\alpha = 3$ of Gallai's conjecture, preprint.
12. X. Zhu, Circular chromatic number: a survey, *Discrete Mathematics*, **229** (2001), 371–410.

A TDI Description of Restricted 2-Matching Polytopes*

Gyula Pap**

Department of Operations Research
Eötvös University, Budapest, Hungary
gyuszko@cs.elte.hu

Abstract. We give a TDI description for a class of polytopes, which corresponds to a restricted 2-matching problem. The perfect matching polytope, triangle-free perfect 2-matching polytope and relaxations of the travelling salesman polytope are members of this class. The paper shows that 2-matching problems for which the unweighted problem was known to be tractable, the weighted is also tractable.

1 Introduction

Let G be an undirected graph, with possible loops or parallel edges. Fix an arbitrary collection \mathcal{H} of some (maybe none) odd length cycles in G . (A *cycle* is the edge set of a 2-regular, connected subgraph, the length of it is given by the number of edges. A loop is regarded as an odd cycle.) An \mathcal{H} -*matching* is the edge set of a subgraph the components of which are single edges and cycles in \mathcal{H} . An \mathcal{H} -*matching* is *perfect* if it covers all nodes. The \mathbb{R}^E -*incidence vector* x_M of an \mathcal{H} -*matching* M is given by $x_M(e) := 1$ if e is in a cycle component of M , $x_M(e) := 2$ if $\{e\}$ is a single edge component of M , all other entries of x_M are zero. In what follows, we will not make a difference between the \mathcal{H} -*matching* and its \mathbb{R}^E -*incidence vector*. The aim of the paper is to give a polyhedral description of the convex hull of perfect \mathcal{H} -*matchings*, denoted by \mathcal{P} .

There are some well-known classes of polytopes arising as \mathcal{P} for a choice of \mathcal{H} . Let us first consider the choice $\mathcal{H} = \emptyset$. In this case, $\mathcal{P} = 2\mathcal{M}$, where \mathcal{M} is the so-called perfect matching polytope. J. Edmonds gave a description of the perfect matching polytope [7], here we mention a stronger theorem due to W. Pulleyblank, J. Edmonds [11] and W.H. Cunningham, A.B. Marsh III [5]. A system of linear inequalities $Ax \leq b$ is called *TDI* or *totally dual integral* if for any integer vector c , the dual program of maximizing cx subject to $Ax \leq b$ has an integer optimum solution y , whenever the optimum value is finite (a definition due to J. Edmonds and R. Giles [8]).

* Research supported by the Hungarian National Foundation for Scientific Research Grant, OTKA T037547.

** The author is a member of the MTA-ELTE Egerváry Research Group (EGRES). Supported by European MCRTN Adonet, Contract Grant No. 504438.

Theorem 1 (Pulleyblank, Edmonds [11]; Cunningham, Marsh [5]). For a graph $G = (V, E)$, the perfect matching polytope is determined by the TDI system

$$x \geq 0 \quad (1)$$

$$x(d(v)) = 1 \quad \text{if } v \in V \quad (2)$$

$$x(E[U]) \leq (|U| - 1)/2 \quad \text{if } U \subseteq V \text{ such that } G[U] \text{ is factor-critical.} \quad (3)$$

Moreover, for any integer vector c there is an integer optimum dual solution y_v, y_U such that $\{E[U] : y_U > 0\}$ is a nice family (defined in Sect. 3).

Consider choosing for \mathcal{H} all odd cycles, except for the cycles of length three: in this case a perfect \mathcal{H} -matching is called a *triangle-free perfect 2-matching*. G. Cornuéjols and W. Pulleyblank gave the following description of the triangle-free perfect 2-matching polytope [2].

Theorem 2 (Cornuéjols, Pulleyblank [2]). For a graph $G = (V, E)$, the triangle-free perfect 2-matching polytope is determined by the TDI system

$$x \geq 0 \quad (4)$$

$$\frac{1}{2}x(d(v)) = 1 \quad \text{if } v \in V \quad (5)$$

$$x(E[a, b, c]) \leq 2 \quad \text{if } \{a, b, c\} \subseteq V(G) \quad (6)$$

For another example, consider the set \mathcal{H} of all odd cycles of length at least 5, and possibly some additional cycles of length 3. In this case we only have a theorem describing \mathcal{P} in case G is simple:

Theorem 3 (Cornuéjols, Pulleyblank [2]). If G is a simple graph, and \mathcal{H} contains all odd cycles of length at least 5, then \mathcal{P} is determined by the following TDI system

$$x \geq 0 \quad (7)$$

$$\frac{1}{2}x(d(v)) = 1 \quad \text{if } v \in V \quad (8)$$

$$x(ab) + x(bc) + x(ca) \leq 2 \quad \text{if } \{ab, bc, ca\} \subseteq E(G), \{ab, bc, ca\} \notin \mathcal{H} \quad (9)$$

In fact, it was an open problem whether there is a good description of this polyhedron if G is not simple. The paper gives a positive answer to a more general problem, but the description given here is slightly more complicated.

The last two systems do not have integer coefficients, but we get integer TDI systems if we replace (5), (8) by $x(d(v)) = 2$ and add $x(E[U]) \leq |U|$ for all sets $U \subseteq V$.

2 The Unweighted Theorem

For a basic introduction to matching theory, see [6]. In the sequel, the most important notion from matching theory is factor-criticality: An undirected graph

is defined to be *factor-critical* if the deletion of any node leaves a graph with a perfect matching.

Let \mathcal{H}' be a set of some odd cycles in $G = (V, E)$ and some nodes in V . If a node is in \mathcal{H}' let us call it a *pseudo-node*. A \mathcal{H}' -matching is a node-disjoint collection of edges, pseudo-nodes and cycles in \mathcal{H}' , which is perfect if it covers V . The size of a \mathcal{H}' -matching is the number of covered nodes. Let $\nu^{\mathcal{H}'}(G)$ denote the maximum size of a \mathcal{H}' -matching. We define the graph G to be \mathcal{H}' -critical if it is factor-critical and there is no perfect \mathcal{H}' -matching in G . An induced subgraph $G[V']$ is \mathcal{H}' -critical if it is factor-critical and there is no perfect \mathcal{H}' -matching in $G[V']$.

We will make use of the following theorem, which is a special case of a theorem in [1] by G. Cornuéjols and D. Hartvigsen.

Theorem 4 (Cornuéjols, Hartvigsen, [1]). *Let $G = (V, E)$ and \mathcal{H}' be as above. There is a perfect \mathcal{H}' -matching in G if and only if there is no set $X \subseteq V$ such that $c_G^{\mathcal{H}'}(X) > |X|$, where $c_G^{\mathcal{H}'}(X)$ denotes the number of \mathcal{H}' -critical components in $G - X$.*

A stronger version is:

Theorem 5 (Cornuéjols, Hartvigsen, [1]). *Let $G = (V, E)$ and \mathcal{H}' be as above. Let $D = D_{\mathcal{H}'}(G)$ be the set of nodes $v \in V$ for which there is an \mathcal{H}' -matching of size $\nu^{\mathcal{H}'}(G)$ which exposes v . Let $A = A_{\mathcal{H}'}(G) = \Gamma_G(D)$, then $\nu^{\mathcal{H}'}(G) = |V| + |A| - c_G^{\mathcal{H}'}(A)$. Furthermore, the components of $G[D]$ are \mathcal{H}' -critical and the other components of $G - A$ have a perfect \mathcal{H}' -matching.*

G. Cornuéjols and W.R. Pulleyblank [3] considered the case when \mathcal{H}' is the set of odd cycles of length at least k , as a relaxation of the Hamiltonian cycle problem. They showed that finding a perfect \mathcal{H}' -matching is polynomially solvable for any k . Complexity issues depend on recognizing \mathcal{H}' -critical graphs. If a factor-critical graph has a perfect \mathcal{H}' -matching then there is a perfect \mathcal{H}' -matching using exactly one member of \mathcal{H}' . The state of the art is that a polynomial-time recognition algorithm could be given in the following cases:

1. There is a number k such that \mathcal{H}' contains all odd cycles longer than k – and maybe some short cycles, too.
2. The members of \mathcal{H}' can be listed in polynomial time. For example each has length less than a fixed number k .

The conclusion of this paper is that in both cases the weighted problem is also tractable. This settles an unpublished conjecture of W. Pulleyblank and M. Loebel [9]. They conjectured that the maximum weight \mathcal{H} -matching problem is polynomially solvable if \mathcal{H} consists of triangles.

3 The Polyhedral Description

A family of sets is called *laminar* if any two of the sets are disjoint or one of them is a subset of the other. A laminar family of edge sets \mathcal{L} is called a *nice*

family if for all sets $F \in \mathcal{L}$, $(V(F), F)$ is factor-critical, and all pairs $F_1, F_2 \in \mathcal{L}$ are either node-disjoint, or F_1 is nice in F_2 , or F_2 is nice in F_1 . We say F_1 is *nice* in F_2 if $F_1 \subseteq F_2$ and $(V(F_2), F_2) - V(F_1)$ has a perfect matching, or equivalently $(V(F_2), F_2)/F_1$ is factor-critical. For example, if $F_1 \subseteq F_2$ and $V(F_1) = V(F_2)$ then F_1 is nice in F_2 . For a set $F \in \mathcal{L}$, let \mathcal{L}_F denote the truncation of \mathcal{L} to subsets of F , including F itself. The truncations to the maximal sets in \mathcal{L} are called *components*.

A pair $\mathcal{F} = (\mathcal{L}, m)$ is called a *nice system* if \mathcal{L} is a nice family, and $m : \mathcal{L} \rightarrow \mathbb{R}$ (with $m(F) \geq 0$ for any set $F \in \mathcal{L}$) is a non-negative “multiplicity function”. Let $\|\mathcal{F}\| := \sum_{\mathcal{L}} m(F)(|V(F)| - 1)$ and $x(\mathcal{F}) := \sum_{\mathcal{L}} m(F)x(F)$ and let $\chi_{\mathcal{F}}(e) := \sum_{\mathcal{L}} m(F)|\{e\} \cap F|$ be the \mathbb{R}^E -characteristic vector of \mathcal{F} . Then $x(\mathcal{F}) = x \cdot \chi_{\mathcal{F}}$ follows. For a set $F \in \mathcal{L}$ let us denote by \mathcal{F}_F the system $(\mathcal{L}_F, m|_{\mathcal{L}_F})$, the components of \mathcal{F} are defined appropriately.

Definition 1. Let \mathcal{F} be a nice system. The inequality $x(\mathcal{F}) \leq \|\mathcal{F}\|$ is called valid if it holds for any \mathcal{H} -matching x ; in this case the system \mathcal{F} will also be called valid.

It is easy to see that \mathcal{F} is valid if and only if its components are valid. Keep in mind that in the definition of validity we considered \mathcal{H} -matchings, while the polytope \mathcal{P} is the convex hull of perfect \mathcal{H} -matchings. However, it is easy to see the following.

Proposition 1. If a nice system \mathcal{F} with one component is not valid, then there is a perfect \mathcal{H} -matching x in $G(\mathcal{F}) = (V(\mathcal{F}), E(\mathcal{F}))$ such that $x(\mathcal{F}) > \|\mathcal{F}\|$.

The main theorem of the paper is the following polyhedral description.

Theorem 6. Let $G = (V, E)$ and \mathcal{H} be as above. Then \mathcal{P} is determined by

$$x \geq 0 \tag{10}$$

$$\frac{1}{2}x(d(v)) = 1 \quad \text{if } v \in V \tag{11}$$

$$x(\mathcal{F}) \leq \|\mathcal{F}\| \quad \text{if } \mathcal{F} \text{ is a valid system.} \tag{12}$$

There is an infinite number of inequalities for valid systems \mathcal{F} . To get a traditional polyhedral description, consider a fixed nice family \mathcal{L} . We define $C_{\mathcal{L}} = \{m : \mathcal{L} \rightarrow \mathbb{R}^+ \text{ such that } (\mathcal{L}, m) \text{ is valid}\}$, then

$$C_{\mathcal{L}} = \{m \geq 0, \tag{13}$$

$$\sum_{F \in \mathcal{L}} m(F)(|V(F)| - 1 - x(F)) \geq 0 \quad \text{for } \mathcal{H}\text{-matchings } x \text{ of } G \tag{14}$$

thus $C_{\mathcal{L}}$ is a polyhedral cone, let $\mathcal{G}_{\mathcal{L}}$ denote a finite generator. The set of solutions of (10)-(12) is the same as of

$$x \geq 0 \tag{15}$$

$$\frac{1}{2}x(d(v)) = 1 \quad \text{if } v \in V \tag{16}$$

$$x(\mathcal{F}) \leq \|\mathcal{F}\| \quad \text{if } \mathcal{F} = (\mathcal{L}, m) \text{ for some } m \in \mathcal{G}_{\mathcal{L}} \tag{17}$$

Notice that we have two complications in comparison with the Theorems 1 and 2. The general class of polytopes \mathcal{P} can not be described by inequalities with 0, ± 1 coefficients, and we also need edge sets of subgraphs which are not induced by a node set. However, we will show in Sect. 6 that there is a TDI description with integer coefficients:

Theorem 7. *The following is a TDI description of \mathcal{P} which has integer coefficients:*

$$x \geq 0 \quad (18)$$

$$x(d(v)) = 2 \quad \text{if } v \in V \quad (19)$$

$$x(E[U]) \leq |U| \quad \text{if } U \subseteq V \quad (20)$$

$$x(\mathcal{F}) \leq \|\mathcal{F}\| \quad \text{if } \mathcal{F} \text{ is a valid system with integer multiplicity.} \quad (21)$$

We get a finite TDI description with integer coefficients if we only put the valid inequalities in (21) for \mathcal{F} corresponding to a Hilbert base of some $C_{\mathcal{L}}$.

4 Decompositions of Valid Inequalities

Definition 2. *A valid system \mathcal{F} is of type (α) if it has one component, m is equal to 1 on the maximal set and zero elsewhere.*

Definition 3. *A valid system \mathcal{F} is of type (β) if it has one component, and there is a perfect \mathcal{H} -matching x^* in $(V(\mathcal{F}), E(\mathcal{F}))$ such that $x^*(\mathcal{F}) = \|\mathcal{F}\|$.*

To proceed with the proof we will need the following technical lemmas, Lemma 4 will be the most important in the next section.

Lemma 1. *If \mathcal{F} is a valid system with one component, $m(F) > 0$ for all sets $F \in \mathcal{L}$, and if $F_1 \in \mathcal{L}$ is not the maximal set, then the truncation \mathcal{F}_{F_1} is not of type (β) .*

Proof. Suppose for contradiction, that x is a perfect \mathcal{H} -matching in $(V(\mathcal{F}_{F_1}), E(\mathcal{F}_{F_1}))$ and $x(\mathcal{F}_{F_1}) = \|\mathcal{F}_{F_1}\|$. Then we construct x' by adding edges with weight 2 such that for all sets $F \in \mathcal{L}$, $F \not\subseteq F_1$ we have $x'(F) = |V(F)| - 1$ or $x'(F) = |V(F)|$, and for sets with $F \supseteq F_1$ only the second alternative holds. Such an x' exists, since \mathcal{L} is a nice family. Then the following calculation gives a contradiction with \mathcal{F} being valid:

$$x'(\mathcal{F}) = x(\mathcal{F}_{F_1}) + \sum_{\mathcal{L} - \mathcal{L}_{F_1}} m(F)x'(F) > \|\mathcal{F}_{F_1}\| + \sum_{\mathcal{L} - \mathcal{L}_{F_1}} m(F)(|V(F)| - 1) = \|\mathcal{F}\|$$

(The inequality holds since m is positive on the maximal set, which is in $\mathcal{L} - \mathcal{L}_{F_1}$.)

Lemma 2. Suppose \mathcal{F} is a valid system with one component and with $m(F) > 0$ for each set $F \in \mathcal{L}$. Then for all sets $F \in \mathcal{L}$ which is not the maximal set in \mathcal{L} ,

- a) the system \mathcal{F}_F is valid, and
- b) for any \mathcal{H} -matching x , the equality $x(\mathcal{F}_F) = ||\mathcal{F}_F||$ implies that $x(F') = |V(F')| - 1$ for all sets $F' \in \mathcal{L}_F$.

Proof. We prove the lemma by induction “from inside to outside”. Suppose we have a set F_1 , such that a) and b) holds for all sets F in $\mathcal{L}_{F_1} - F_1$.

Suppose F_1 is not the maximal set, and x is an \mathcal{H} -matching such that $x(\mathcal{F}_{F_1}) \geq ||\mathcal{F}_{F_1}||$. Then $x(F_1) = |V(F_1)|$ leads to a contradiction, since an \mathcal{H} -matching x' could be constructed as in Lemma 2 such that $x'(\mathcal{F}) > ||\mathcal{F}||$. Thus $x(F_1) \leq |V(F_1)| - 1$ and then we get statement a) for F_1 by

$$x(\mathcal{F}_{F_1}) = x(F_1) + \sum x(\mathcal{F}_{D_i}) \leq |V(F_1)| - 1 + \sum ||\mathcal{F}_{D_i}|| = ||\mathcal{F}_{F_1}||$$

where D_i are the maximal sets in $\mathcal{L}_{F_1} - F_1$.

If for x we have $x(\mathcal{F}_{F_1}) = ||\mathcal{F}_{F_1}||$, then all equalities $x(\mathcal{F}_{D_i}) = ||\mathcal{F}_{D_i}||$ must hold. Then by induction we get that for each set $F' \in \mathcal{L}_{F_1} - F_1$ the equality $x(F') = |V(F')| - 1$ holds. This also implies $x(F_1) = |V(F_1)| - 1$ and then b) holds for F_1 .

Notice, that a) also holds for the maximal set, but b) does not hold necessarily.

Lemma 3. Suppose \mathcal{F} is a valid system with one component, with $m(F) > 0$ for each set $F \in \mathcal{L}$, and \mathcal{F} is not of type (β) . Then for each set $F \in \mathcal{L}$ there is a multiplicity m on \mathcal{L}_F such that for the systems $\widehat{\mathcal{F}}_F := (\mathcal{L}_F, m_F)$

1. $\widehat{\mathcal{F}}_F \sim (\alpha)$, or
2. $\widehat{\mathcal{F}}_F \sim (\beta)$ and $m_F(F') > 0$ for all sets $F' \in \mathcal{L}_F$,

and there are coefficients $\lambda_F > 0$ such that

$$\chi_{\mathcal{F}} = \sum_{F \in \mathcal{L}} \lambda_F \cdot \chi_{\widehat{\mathcal{F}}_F}. \quad (22)$$

Proof. The proof goes by induction on $|\mathcal{L}|$. For $|\mathcal{L}| = 1$ the statement holds, since \mathcal{F} is a positive multiple of a system of type (α) .

Let F_0 be the maximal set in \mathcal{L} . By Lemma 2, all truncations of \mathcal{F} are valid, and by Lemma 1 no proper truncation of \mathcal{F} is of type (β) . Thus by induction one can give for each set $F \in \mathcal{L} - F_0$ a system $\widehat{\mathcal{F}}_F = (\mathcal{L}_F, m_F)$ as in 1. or 2. and coefficients $\lambda_F > 0$ such that

$$\chi_{\mathcal{F}} - m(F_0) \cdot \chi_{F_0} = \sum_{F \in \mathcal{L} - F_0} \lambda_F \cdot \chi_{\widehat{\mathcal{F}}_F} \quad (23)$$

holds. There are two cases. First, if there is no perfect \mathcal{H} -matching in $(V(F_0), F_0)$ then $m_{F_0}(F_0) := 1$ and $m_{F_0}(F) := 0$ (for $F \neq F_0$) and $\lambda_{F_0} := m_{\mathcal{F}}(F_0) > 0$ give equality in (22).

Second, if there is at least one perfect \mathcal{H} -matching in $(V(F_0), F_0)$. Since \mathcal{F} is not of type (β) , for each perfect \mathcal{H} -matching x in $(V(F_0), F_0)$ we have $x(\mathcal{F}) < \|\mathcal{F}\|$. For a number $t > 0$ we let $\mathcal{F}^t := (\mathcal{L}, m^t)$ where $m^t(F) = m(F)$ if $F \neq F_0$, and $m^t(F_0) = t$. Let $t_0 := m(F_0)$, then $\mathcal{F}^{t_0} = \mathcal{F}$. There is a uniquely defined number T such that \mathcal{F}^T is of type (β) . Then $T > t_0$ holds, let $\widehat{\mathcal{F}}_{F_0} := \mathcal{F}^T$ and $\lambda'_{F_0} := t_0/T$, $\lambda'_F := (1 - t_0/T)\lambda_F$. This gives the desired decomposition.

Lemma 4. *Suppose \mathcal{F} is a valid system with one component, with $m(F) > 0$ for each set $F \in \mathcal{L}$, and \mathcal{F} is not of type (β) . If x is an \mathcal{H} -matching such that $x(\mathcal{F}) = \|\mathcal{F}\|$, then $x(F') = |V(F')| - 1$ for all sets $F' \in \mathcal{L}$.*

Proof. Take the decomposition $\chi_{\mathcal{F}} = \sum_{F \in \mathcal{L}} \lambda_F \cdot \chi_{\widehat{\mathcal{F}}_F}$ in (22), also $\|\mathcal{F}\| = \sum_{F \in \mathcal{L}} \lambda_F \cdot \|\widehat{\mathcal{F}}_F\|$ holds. Since $\lambda_F > 0$, this implies $x(\widehat{\mathcal{F}}_F) = \|\widehat{\mathcal{F}}_F\|$.

Suppose for $F' \in \mathcal{L}$ we have $x(F'') = |V(F'')| - 1$ for all sets $F'' \in \mathcal{L}$, $F'' \subseteq F'$. Then $x(F') = |V(F')| - 1$ follows from $x(\widehat{\mathcal{F}}_F) = \|\widehat{\mathcal{F}}_F\|$.

Lemma 3 implies that $\mathcal{G}_{\mathcal{L}}$ can be chosen such that each member is a nice system of type (α) or (β) . The valid systems of type (α) are determined by an edge set. The valid systems \mathcal{F} of type (β) can be described as follows: Consider a fixed nice system \mathcal{L} , and a perfect \mathcal{H} -matching x^* in $(V(\mathcal{L}), \mathcal{L})$. We define $C_{\mathcal{L}, x^*} = \{m : \mathcal{L} \rightarrow \mathbb{R}^+ \text{ such that } (\mathcal{L}, m) \text{ is valid, and } x^*(\mathcal{L}, m) = \|(\mathcal{L}, m)\|\}$. Then

$$C_{\mathcal{L}, x^*} = \{m \geq 0, \quad (24)$$

$$\sum_{F \in \mathcal{L}} m(F) (|V(F)| - 1 - x^*(F)) = 0, \quad (25)$$

$$\sum_{F \in \mathcal{L}} m(F) (|V(F)| - 1 - x(F)) \geq 0 \text{ for } \mathcal{H}\text{-matchings } x \text{ of } G \quad (26)$$

thus $C_{\mathcal{L}, x^*}$ is a polyhedral cone. Let $\mathcal{G}(\alpha)$ and $\mathcal{G}(\beta)$ be the set of valid systems we get from the finite generators of these cones. Then Theorem 6 implies that \mathcal{P} is determined by

$$x \geq 0 \quad (27)$$

$$\frac{1}{2}x(d(v)) = 1 \quad \text{if } v \in V \quad (28)$$

$$x(\mathcal{F}) \leq \|\mathcal{F}\| \quad \text{if } \mathcal{F} \in \mathcal{G}(\alpha) \cup \mathcal{G}(\beta). \quad (29)$$

5 Proof by a Primal-Dual Method

Instead of taking the dual program of (10)-(12), we consider a bunch of linear programs each of which corresponds to a nice family. There is a large, but finite number of nice families in any graph G . For a nice family \mathcal{L} , consider the following linear program in $\mathbb{R}^{V \cup \mathcal{L}}$:

$$\min \sum_{v \in V} y_v + \sum_{F \in \mathcal{L}} m_F \cdot (|V(F)| - 1) \quad (30)$$

$$\frac{1}{2}y_u + \frac{1}{2}y_v + \sum_{uv \in F \in \mathcal{L}} m(F) \geq c_{uv} \quad \text{for } uv \in E - E(\mathcal{L}) \quad (31)$$

$$\frac{1}{2}y_u + \frac{1}{2}y_v + \sum_{uv \in F \in \mathcal{L}} m(F) = c_{uv} \quad \text{for } uv \in E(\mathcal{L}) \quad (32)$$

$$m_F \geq 0 \quad \text{for } F \in \mathcal{L} \quad (33)$$

$$\sum_{F \in \mathcal{L}} m(F) (|V(F)| - 1 - x(F)) \geq 0 \quad \text{for } \mathcal{H}\text{-matchings } x \text{ of } G \quad (34)$$

Notice that the part (33)-(34) is equivalent to (\mathcal{L}, m) being valid, and the objective in (30) is equal to $\sum_{v \in V} y_v + \|(\mathcal{L}, m)\|$. Thus, for some solution (y, m) of (31)-(34) one can easily construct a dual solution of (10)-(12) of the same objective value. We abbreviate the objective in (30) by $(y, m) \cdot b$.

Suppose there is at least one perfect \mathcal{H} -matching x in G , then $c \cdot x$ is a lower bound on $(y, m) \cdot b$ for a solution of some system (31)-(34). Choose \mathcal{L} such that the minimum $(y, m) \cdot b$ in (30) is minimal, let (y, m) be a minimizing vector. The minimum is also attained by a pair $\mathcal{L}, (y, m)$ such that $m(F) > 0$ holds for all sets $F \in \mathcal{L}$ (the other sets can be eliminated from the laminar family). We are aiming to prove the existence of a perfect \mathcal{H} -matching x such that $c \cdot x = (y, m) \cdot b$ for which we need these complementary slackness conditions:

$$x_{uv} > 0 \text{ implies } \frac{1}{2}y_u + \frac{1}{2}y_v + \sum_{uv \in F \in \mathcal{L}} m(F) = c_{uv} \quad (35)$$

$$x(\mathcal{L}, m) = \|(\mathcal{L}, m)\| \quad (36)$$

Let $E_{(y, m)}$ denote the set of *tight edges*, the edges uv for which equality holds in (31) or (32). Let $G_{(y, m)} = (V, E_{(y, m)})$, and let $\overline{G}_{(y, m)} = (\overline{V}, \overline{E}_{(y, m)})$ be the graph we get from $G_{(y, m)}$ by contracting all edges in $E[\mathcal{L}]$. Notice that this way we could get a lot of loops or parallel edges, and it is of great importance to keep them. For a set $\overline{U} \subseteq \overline{V}$ let U denote set of the corresponding nodes in V .

Let $\mathcal{L}_{(\beta)}$ denote the union of those components of \mathcal{L} which is of type (β) . We call a node in \overline{V} a pseudo-node if it corresponds to a component of $\mathcal{L}_{(\beta)}$.

$\mathcal{M}_{(y, m)} := \{x : x \text{ is an } \mathcal{H}\text{-matching such that}$

$$\text{for all sets } F \in \mathcal{L} \text{ equation } x(F) = |V(F)| - 1 \text{ holds}\} \quad (37)$$

$$\mathcal{H}_{(y, m)}^* := \{\text{odd cycles, which appear in some } x \in \mathcal{M}_{(y, m)}\}. \quad (38)$$

Let $\overline{\mathcal{H}}_{(y, m)}$ denote the set of the pseudo-nodes, plus the odd cycles in $\overline{G}_{(y, m)}$ we get from $\mathcal{H}_{(y, m)}^*$ by contracting the edges in $E(\mathcal{L})$. It follows from the definition of $\mathcal{M}_{(y, m)}$, that the contraction produces only cycles. For an $\mathcal{H}_{(y, m)}^*$ -matching x in $\mathcal{M}_{(y, m)}$ let x/\mathcal{L} denote the resulting $\overline{\mathcal{H}}_{(y, m)}$ -matching in $\overline{G}_{(y, m)}$ after contracting the edges in $E(\mathcal{L})$.

Suppose there is a perfect $\overline{\mathcal{H}}_{(y,m)}$ -matching \overline{x} in $\overline{G}_{(y,m)}$. In this case we construct a perfect \mathcal{H} -matching x in G such that $(y, m) \cdot b = c \cdot x$, as follows. Expand a contracted node v : if in \overline{x} a node v is covered by an edge then use a near perfect matching, if in \overline{x} a node v is covered by a cycle then use the corresponding \mathcal{H} -matching in G from the definition of $\mathcal{M}_{(y,m)}$. If a pseudo-node is exposed by \overline{x} then expand it with the perfect \mathcal{H} -matching x^* from the definition of (β) . Then (35) follows from the fact that the perfect \mathcal{H} -matching x has only tight edges and (36) follows from the definitions of (β) and $\overline{\mathcal{H}}_{(y,m)}$.

If there is no perfect $\overline{\mathcal{H}}_{(y,m)}$ -matching in $\overline{G}_{(y,m)}$, then by Theorem 4 there is a set $\overline{X} \subseteq \overline{V}$ such that the number of $\overline{\mathcal{H}}_{(y,m)}$ -critical components in $\overline{G}_{(y,m)} - \overline{X}$ is strictly greater than $|\overline{X}|$.

Let $\mathcal{F} := \mathcal{F}_{(y,m)}$ be the system with nice family \mathcal{L} and multiplicity m . For $\overline{K} \subseteq \overline{V}$ we define $\mathcal{F}_{E[\overline{K}]}$ to be the truncation of \mathcal{F} to the subsets of $E[\overline{K}]$.

Lemma 5. *If $\overline{G}_{(y,m)}[\overline{K}]$ is an $\overline{\mathcal{H}}_{(y,m)}$ -critical component in $\overline{G}_{(y,m)} - \overline{X}$ for some $\overline{K} \subseteq \overline{V}$, then either*

- a) $G_{(y,m)}[K]$ is \mathcal{H} -critical, or
- b) for each perfect \mathcal{H} -matching x in $G_{(y,m)}[K]$ we have $x(\mathcal{F}_{E[K]}) < \|\mathcal{F}_{E[K]}\|$.

Proof. Since $\overline{G}_{(y,m)}[\overline{K}]$ is $\overline{\mathcal{H}}_{(y,m)}$ -critical, all components of $\mathcal{L}_{(\beta)}$ are node-disjoint from K . For each \mathcal{H} -matching x in $G_{(y,m)}[K]$ we have $x(\mathcal{F}_{E[K]}) \leq \|\mathcal{F}_{E[K]}\|$, since \mathcal{F} is valid. Suppose for contradiction, that x is a perfect \mathcal{H} -matching in $G_{(y,m)}[K]$ such that $x(\mathcal{F}_{E[K]}) = \|\mathcal{F}_{E[K]}\|$. By Lemma 4 we get that for each set $F \in \mathcal{F}_{E[K]}$ the equation $x(F) = |V(F)| - 1$ holds. Then it is easy to see, that x/\mathcal{L} is a perfect $\overline{\mathcal{H}}_{(y,m)}$ -matching in $\overline{G}_{(y,m)}[\overline{K}]$, a contradiction.

Now, we have a set \overline{X} at hand, which enables us to construct a solution of (31)-(34) for some other laminar family \mathcal{L}' as follows. Let \overline{K}_i ($i = 1, \dots, k$) be the set of $\overline{\mathcal{H}}_{(y,m)}$ -critical components in $\overline{G}_{(y,m)} - \overline{X}$, here $k > |\overline{X}|$; furthermore we define $F_i := E_{(y,m)}[K_i]$. Let X_1, \dots, X_l denote the maximal sets (of edges) in \mathcal{L} that correspond to a node in \overline{X} . Let $\mathcal{L}' := \mathcal{L} \cup \{F_1, \dots, F_k\}$, and

$$\begin{aligned} m'(F_i) &:= m(F_i) + \varepsilon && \text{if } F_i \in \mathcal{L} \\ m'(F_i) &:= \varepsilon && \text{if } F_i \notin \mathcal{L} \\ m'(X_i) &:= m(X_i) - \varepsilon && \text{for } i = 1, \dots, l \\ m'(F) &:= m(F) && \text{otherwise} \\ y'(v) &:= y(v) && \text{if } v \in V - X - \cup K_i \\ y'(v) &:= y(v) + \varepsilon && \text{if } v \in X \\ y'(v) &:= y(v) - \varepsilon && \text{if } v \in \cup K_i \end{aligned}$$

This dual change does not change $\frac{1}{2}y_u + \frac{1}{2}y_v + \sum_{uv \in F \in \mathcal{L}} m(F)$ for $uv \in E(\mathcal{L}')$, and does not decrease $\frac{1}{2}y_u + \frac{1}{2}y_v + \sum_{uv \in F \in \mathcal{L}} m(F)$ for $uv \in E_{(y,m)}$, and m was strictly positive. By (31)-(33) we get a positive upper bound on ε from edges in $E - E_{(y,m)}$ joining a node in $\cup K_i$ to a node in $V - X$, and the minimum of m on the sets X_i .

We need to prove that we can choose $\varepsilon > 0$ to be small enough to get a solution of (34) for \mathcal{L}' . It is enough to prove that if \mathcal{L}'' is a component of \mathcal{L}' , then $\sum_{F \in \mathcal{L}''} m'(F) (|V(F)| - 1 - x(F)) \geq 0$ holds for any \mathcal{H} -matching x of G . The components of \mathcal{F}' are \mathcal{F}'_{F_i} and \mathcal{F}'_{X_i} , or are identical with a component of \mathcal{F} , we only have to check the first two cases. Thus we need to prove, that the systems \mathcal{F}'_{X_i} and \mathcal{F}'_{F_i} are valid if ε is small enough.

Let X_i^j be the maximal sets in $\mathcal{L}_{X_i} - \{X_i\}$. Since

$$\chi_{\mathcal{F}'_{X_i}} = \left(1 - \frac{\varepsilon}{m(X_i)}\right) \cdot \chi_{\mathcal{F}_{X_i}} + \frac{\varepsilon}{m(X_i)} \cdot \sum_j \chi_{\mathcal{F}_{X_i^j}}$$

and by Lemma 2, the systems \mathcal{F}_{X_i} and $\mathcal{F}_{X_i^j}$ are valid, we are done.

For a component \mathcal{F}'_{F_i} and an \mathcal{H} -matching x we need:

$$\begin{aligned} 0 \leq \sum_{F \in \mathcal{L}'_{F_i}} m'(F) (|V(F)| - 1 - x(F)) &= \\ &= \sum_{F \in \mathcal{L}'_{F_i}} m(F) (|V(F)| - 1 - x(F)) + \varepsilon \cdot (|V(F_i)| - 1 - x(F_i)) = \\ &= ||\mathcal{F}_{F_i}|| - x(\mathcal{F}_{F_i}) + \varepsilon \cdot (|V(F_i)| - 1 - x(F_i)). \end{aligned}$$

The upper bound on ε is the minimum of the fraction $(||\mathcal{F}_{F_i}|| - x(\mathcal{F}_{F_i})) / (x(F_i) - |V(F_i)| + 1)$ on \mathcal{H} -matchings x for which $x(F_i) - |V(F_i)| + 1$ is positive. It is easy to see that $x(F_i) - |V(F_i)| + 1$ is positive if and only if x is a perfect \mathcal{H} -matching in $(V(F_i), F_i) = G_{(y,m)}[K_i]$. By Lemma 5, for such an x we have $||\mathcal{F}_{F_i}|| - x(\mathcal{F}_{F_i}) > 0$.

Thus if we choose $\varepsilon > 0$ to be small enough, we get a solution of (31)-(34) for the laminar system \mathcal{L}' . Then $(y', m')b' = (y, m)b - (k - |\overline{X}|)\varepsilon < (y, m)b$ contradicts the choice of (y, m) . This finishes the proof of the description (10)-(12) of \mathcal{P} . In fact, we have proved a little bit more:

Theorem 8. *For each vector c , there is a nice family \mathcal{L} such that the optimum value of (30)-(34) is equal to $\max_{x \in \mathcal{P}} cx$.*

6 TDI Descriptions and Algorithms

Up to this point we have not discussed the complexity of the notions introduced. The following lemma is essential, we skip the proof in this paper.

Lemma 6 ([10]). *Suppose $\mathcal{F} = (\mathcal{L}, m)$ is a valid system with one component and there is at least one perfect \mathcal{H} -matching in $G(\mathcal{F}) = (V(\mathcal{F}), E(\mathcal{F}))$. Then the maximum $M(\mathcal{F}) := \max\{x(\mathcal{F}) : x \text{ is a perfect } \mathcal{H}\text{-matching in } G(\mathcal{F})\}$ is attained by a perfect \mathcal{H} -matching x which has exactly one odd cycle.*

Here we sketch an algorithm to check whether a nice system with one component \mathcal{F} is valid and if so, to find the maximum $M(\mathcal{F})$. Let $M'(\mathcal{F})$ denote $\max\{x(\mathcal{F}) : x \text{ is a perfect } \mathcal{H}\text{-matching in } G(\mathcal{F}) \text{ with exactly one odd cycle}\}$, the statement of the Lemma 6 is that if \mathcal{F} is valid, then $M(\mathcal{F}) = M'(\mathcal{F})$.

Lemma 7. *In cases 1 and 2 of Sect. 2 the maximum $M'(\mathcal{F})$ can be determined in polynomial time for any nice system.*

Lemma 8. *A nice system $\mathcal{F} = (\mathcal{L}, m)$ with one component and maximal set F_0 is valid if and only if its proper truncations are valid and $M'(\mathcal{F}) \leq \|\mathcal{F}\|$.*

Lemma 7 can be proved by a simple reduction to weighted perfect matching using a principle from [3], while Lemma 8 easily follows from the definitions and Lemma 6. The correctness of the following algorithm follows from these lemmas. By Proposition 1, this algorithm is in fact a separation algorithm for $\mathcal{C}_{\mathcal{L}}$.

Algorithm 1. *Checking validity and finding $M(\mathcal{F})$.*

1. Determine $M'(\mathcal{F}_F)$ for each set $F \in \mathcal{F}$.
2. \mathcal{F} is valid if and only if $M'(\mathcal{F}_F) \leq \|\mathcal{F}_F\|$ holds for each $F \in \mathcal{F}$.
3. If \mathcal{F} is valid, then $M(\mathcal{F}) = M'(\mathcal{F})$.

Now we prove the following statement on dual integrality.

Theorem 9. *For each integer vector c , there is a nice family \mathcal{L} such that the optimum value of (30)-(34) is equal to $\max_{x \in \mathcal{P}} cx$ and is attained by an integer vector (y, m) .*

The proof goes by carefully reading Sect. 5, using the dual change for an algorithm.

We start with \mathcal{L} empty and (y, m) some integer solution of (30)-(34). Then we iteratively change \mathcal{L} and (y, m) as written in Sect. 5: we find the sets X and K_i , change \mathcal{L} and (y, m) , and if an entry of m' is 0 then we eliminate the set from \mathcal{L}' . The number ε is the only data not well-defined in Sect. 5, using the notation M and $c'(uv) = c(uv) - \frac{1}{2}y_u - \frac{1}{2}y_v - \sum_{uv \in F \in \mathcal{L}} m(F)$ we have

$$\begin{aligned} \varepsilon &= \min\{2c'(uv) && \text{for } uv \notin E_{(y,m)}, u \in \cup K_i, v \in V - X - \cup K_i \\ c'(uv) && \text{for } uv \notin E_{(y,m)}, u, v \in \cup K_i \\ m(X_i) && \text{for } i = 1, \dots, l \\ \|\mathcal{F}_{F_i}\| - M(\mathcal{F}_{F_i}) && \text{for } i = 1, \dots, k. \end{aligned}$$

If this minimum was taken over an emptyset, then it is easy to see that set X is as in Theorem 4 which verifies that there is no perfect \mathcal{H} -matching. Otherwise if there is a perfect \mathcal{H} -matching in G , but no perfect $\overline{\mathcal{H}}_{(y,m)}$ -matching in $\overline{G}_{(y,m)}$, to maintain integrality of (y, m) let us choose X and K_i according to the following rule. Consider the sets $D = D_{\overline{\mathcal{H}}_{(y,m)}}(\overline{G}_{(y,m)})$ and $A = A_{\overline{\mathcal{H}}_{(y,m)}}(\overline{G}_{(y,m)})$. Let $\overline{G}_{(y,m)}[Z]$ be a connected component of $\overline{G}_{(y,m)}[D \cup A]$ for $Z \subseteq D \cup A$. Let $\overline{X} := Z \cap A$ and let \overline{K}_i be the components of $Z \cap D$.

If (y, m) is integer, then the parity of y is the same on each node of a component of $G_{(y,m)}$. Thus y_v has the same parity for the nodes v in K_i and X , we get that ε is integer. After the dual change, (y', m') is an integer vector. A dual change does not increase the deficit $\text{def}_{(y,m)} := |\overline{V}| - \nu_{\overline{\mathcal{H}}_{(y,m)}}(\overline{G}_{(y,m)})$. As long as

this deficit stays on the same value, a dual change does not decrease the number $d_{(y,m)}$ of nodes of V corresponding to a node in $D_{\overline{\mathcal{H}}_{(y,m)}}(\overline{G}_{(y,m)})$. As long as $d_{(y,m)}$ and $\text{def}_{(y,m)}$ stay on the same value, each dual change must be with ε determined by some $m(X_i)$. Then the set X_i will be eliminated, the number of sets in $\mathcal{L}_{E[A]}$ decreases. The number of dual changes needed is at most $|V|^2 \cdot |E|$. This provides an algorithm for finding a maximum weight perfect \mathcal{H} -matching together with an integer dual optimum, and proves that (15)-(17) is TDI. The algorithm is polynomial if we have an oracle to solve the unweighted problem in each contracted graph, and we have an oracle to find the maximum $M(\mathcal{F})$ for any nice family with one component. In cases 1 and 2 of Sect. 2 the set $\underline{\mathcal{H}}_{(y,m)}$ is also of the same case with the same k , which gives us the first oracle. The second oracle is given by Lemma 7 and Algorithm 1.

We will use an elementary technique to derive a TDI description with integer coefficients.

Proof. (of Theorem 7) For some fixed integer vector c , consider an integer optimum solution (y, m) of (30)-(34). Let $y'_v := \lfloor \frac{1}{2}y_v \rfloor$, let U_1 be the set of nodes v with y_v odd, and $z'_U := 1$ if $U = U_1$, otherwise $z'_U := 0$. Let ω' be a 0-1 vector with the only 1 in the entry for $\mathcal{F} = (\mathcal{L}, m)$. Then (y', z', ω') is an optimum dual solution of (18)-(21).

Acknowledgment

The author thanks Tamás Király and Márton Makai for helpful discussions on the topic.

References

1. Cornuéjols, G., Hartvigsen, D.: An extension of matching theory. *J. Combin. Theory Ser. B* **40** (1986) 285–296
2. Cornuéjols, G., Pulleyblank, W.R.: A matching problem with side conditions. *Discrete Mathematics*, **29** (1980) 135–159
3. Cornuéjols G., Pulleyblank, W.R.: Critical graphs, matchings and tours or a hierarchy of the travelling salesman problem. *Combinatorica* **3(1)** (1983) 35–52
4. Cornuéjols, G., Pulleyblank, W.R.: The travelling salesman polytope and $\{0, 2\}$ -matchings. *Annals of Discrete Mathematics* **16** (1982) 27–55
5. Cunningham, W.H., Marsh III, A.B.: A primal algorithm for optimum matching. *Math. Programming Stud.* **8** (1978) 50–72
6. Lovász L., Plummer, M.D.: *Matching Theory*. Akadémiai Kiadó, Budapest, 1986.
7. Edmonds, J.: Maximum matching and a polyhedron with 0,1 vertices. *Journal of Research National Bureau of Standards Section B* **69** (1965) 125–130
8. Edmonds, J., Giles, R.: A min-max relation for submodular functions on graphs. *Studies in Integer Programming (Proceedings Workshop on Integer Programming, Bonn, 1975; P.L. Hammer, E.L. Johnson, B.H. Korte, G.L. Nemhauser, eds. Annals of Discrete Mathematics 1)*, North Holland, Amsterdam, (1977) 185–204
9. Loebel, M.: personal communication, 2004.

10. Pap, P.: manuscript, 2004.
11. Pulleyblank, W.R., Edmonds, J.: Facets of 1-matching polyhedra. in: Hypergraph Seminar (Proceedings Working Seminar on Hypergraphs, Columbus, Ohio, 1972; C. Berge, D. Ray-Chaudhouri, eds.), Springer, Berlin (1974) 214–242

Enumerating Minimal Dicuts and Strongly Connected Subgraphs and Related Geometric Problems*

E. Boros¹, K. Elbassioni¹, V. Gurvich¹, and L. Khachiyan²

¹ RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway NJ 08854-8003;
{boros,elbassio,gurvich}@rutcor.rutgers.edu

² Department of Computer Science, Rutgers University, 110 Frelinghuysen Road,
Piscataway NJ 08854-8003;
leonid@cs.rutgers.edu

Abstract. We consider the problems of enumerating all minimal strongly connected subgraphs and all minimal dicuts of a given directed graph $G = (V, E)$. We show that the first of these problems can be solved in incremental polynomial time, while the second problem is NP-hard: given a collection of minimal dicuts for G , it is NP-complete to tell whether it can be extended. The latter result implies, in particular, that for a given set of points $\mathcal{A} \subseteq \mathbb{R}^n$, it is NP-hard to generate all maximal subsets of \mathcal{A} contained in a closed half-space through the origin. We also discuss the enumeration of all minimal subsets of \mathcal{A} whose convex hull contains the origin as an interior point, and show that this problem includes as a special case the well-known hypergraph transversal problem.

1 Introduction

Let V be a finite set of vertices and $G = (V, E)$ be a *strongly connected* digraph on V with arc set $E \subseteq V \times V$. A minimal strongly connected subgraph of G is a minimal subset of arcs $X \subseteq E$ such that the digraph (V, X) is strongly connected. A *minimal directed cut*, or a *dicut* in G is a minimal subset of arcs the removal of which leaves a non-strongly connected digraph. Given two specified vertices $s, t \in V$, a minimal (s, t) -*dicut* is a minimal subset of edges whose removal leaves no directed path from s to t . The analogous notions for undirected graphs correspond respectively to spanning trees, minimal cuts, and s, t -cuts.

These notions play an important role in network reliability, where edges or arcs represent communication or transportation links, which may work or fail independently, and where the main problem is to determine the probability that the network is working, based on the individual edge/arc failure probabilities. It turns out that such network reliability computations require in the general

* This research was supported by the National Science Foundation (Grant IIS-0118635). The second and third authors are also grateful for the partial support by DIMACS, the National Science Foundation's Center for Discrete Mathematics and Theoretical Computer Science.

case the list of minimal cuts, dicuts, s, t -cuts, etc., depending on the type of connectivity the network is ought to maintain (i.e., all-terminal, two-terminal, strong, etc.), see e.g., [1,3,7,16].

It is quite easy to see that the number of spanning trees, cuts, s, t -paths, s, t -cuts, etc. is, in general, exponential in the size of the graph. For this reason the efficiency of their generation is measured in both the input and output sizes, e.g., we shall talk about the complexity of generation "per cut".

Given a strongly connected digraph $G = (V, E)$, let us denote by \mathcal{F}_{sc} the family of minimal strongly connected subgraphs (V, X) of G . Then the *dual* family \mathcal{F}_{sc}^d is the set of minimal dicuts of G :

$$\mathcal{F}_{sc}^d = \{X \subseteq E : X \text{ is a minimal transversal of } \mathcal{F}_{sc}\},$$

where $X \subseteq E$ is a *transversal* of \mathcal{F}_{sc} if and only if $X \cap Y \neq \emptyset$ for all $Y \in \mathcal{F}_{sc}$.

In this paper, we shall consider the problem of listing incrementally all the elements of the families \mathcal{F}_{sc} and \mathcal{F}_{sc}^d . Enumeration algorithms for listing minimal families \mathcal{F}_π of subgraphs satisfying a number of monotone properties π are well known. For instance, it is known [18] that the problems of listing all minimal cuts or all spanning trees of an undirected graph $G = (V, E)$ can be solved with delay $O(|E|)$ per generated cut or spanning tree. It is also known (see e.g., [8,12,17]) that all minimal (s, t) -cuts or (s, t) -paths, can be listed with delay $O(|E|)$ per cut or path, both in the directed and undirected cases. Furthermore, if $\pi(X)$ is the property that the subgraph (V, X) of a directed graph $G = (V, E)$ contains a directed cycle, then \mathcal{F}_π is the family of minimal directed circuits of G , while \mathcal{F}_π^d consists of all minimal feedback arc sets of G . Both of these families can be generated with polynomial delay per output element, see e.g. [20].

It is quite remarkable that in all these cases both the family \mathcal{F}_π and its dual \mathcal{F}_π^d can be generated efficiently, unlike in case of many other monotone families, [6,14]. In this paper we focus on a further case relevant to reliability theory, when this symmetry is broken. Specifically, we show the problem of incrementally generating all minimal dicuts of a given strongly connected directed graph is NP-hard.

Theorem 1. *Given a strongly connected directed graph $G = (V, E)$ and a partial list $\mathcal{X} \subseteq \mathcal{F}_{sc}^d$ of minimal dicuts of G , it is NP-hard to determine if the given list is complete, i.e. if $\mathcal{F}_{sc}^d = \mathcal{X}$.*

We show also that on the contrary, listing minimal strongly connected subgraphs can be done efficiently.

Theorem 2. *Given a strongly connected directed graph $G = (V, E)$, all minimal strongly connected subgraphs of G can be listed in incremental polynomial time.*

We prove Theorems 1 and 2 in Sections 3 and 4 respectively. In the next section, we discuss some geometric generalizations of these problems related to the enumeration of all minimal subsets of a given set of points $\mathcal{A} \subseteq \mathbb{R}^n$ whose convex hull contains the origin as an interior point, and all maximal subsets of

\mathcal{A} whose convex hull does not contain the origin as an interior point. We will illustrate that the former problem is NP-hard, while the latter is at least as hard as the well-known hypergraph transversal problem, and discuss the relation of these two problems to the still open problem of vertex enumeration.

2 Some Related Geometric Problems

Let $\mathcal{A} \subseteq \mathbb{R}^n$ be a given subset of vectors in \mathbb{R}^n . Fix a point $z \in \mathbb{R}^n$, say $z = \mathbf{0}$, and consider the following four geometric objects:

- A *simplex* is a minimal subset $X \subseteq \mathcal{A}$ of vectors containing z in its convex hull: $z \in \text{conv}(X)$.
- An *anti-simplex* is a maximal subset $X \subseteq \mathcal{A}$ of vectors not containing z in its convex hull: $z \notin \text{conv}(X)$.
- A *body* is a minimal (full-dimensional) subset $X \subseteq \mathcal{A}$ of vectors containing z in the interior of its convex hull: $z \in \text{int conv}(X)$.
- An *anti-body* is a maximal subset $X \subseteq \mathcal{A}$ of vectors not containing z in the interior of its convex hull: $z \notin \text{int conv}(X)$.

Equivalently, a simplex (body) is a minimal collection of vectors not contained in an *open* (*closed*) half-space through z . An anti-simplex (anti-body) is a maximal collection of vectors contained in an open (*closed*) half space. It is known that $|X| \leq n + 1$ for any simplex $X \subseteq \mathcal{A}$ and that $n + 1 \leq |X| \leq 2n$ for any body $X \subseteq \mathcal{A}$.

For a given point set \mathcal{A} , denote respectively by $\mathcal{S}(\mathcal{A})$ and $\mathcal{B}(\mathcal{A})$ the families of simplices and bodies of \mathcal{A} with respect to the origin $z = \mathbf{0}$. Then it is clear that the complementary families $\mathcal{S}(\mathcal{A})^{dc}$ and $\mathcal{B}(\mathcal{A})^{dc}$ of their duals (where for a family $\mathcal{F} \subseteq 2^{\mathcal{A}}$, the complementary family $\mathcal{F}^c = \{\mathcal{A} \setminus X \mid X \in \mathcal{F}\}$) are respectively the families of anti-simplices and anti-bodies of \mathcal{A} .

Let $A \in \mathbb{R}^{m \times n}$, where $m = |\mathcal{A}|$, be the matrix whose rows are the points of \mathcal{A} . It follows from the above definitions that simplices and anti-simplices are in one-to-one correspondence respectively with the *minimal infeasible* and *maximal feasible* subsystems of the linear system of inequalities:

$$Ax \geq \mathbf{e}, \quad x \in \mathbb{R}^n \tag{1}$$

where $e \in \mathbb{R}^m$ is the m -dimensional vector of all ones. Similarly, it follows that bodies and anti-bodies correspond respectively to the minimal infeasible and maximal feasible subsystems of the system:

$$Ax \geq \mathbf{0}, \quad x \neq \mathbf{0}. \tag{2}$$

As a special case of the above problems, let $G = (V, E)$ be a directed graph, and let $\mathcal{A} \subseteq \{-1, 0, 1\}^V$ be the set of incidence vectors corresponding to the arc-set E , i.e. $\mathcal{A} = \{\chi(a, b) : (a, b) \in E\}$, where $\chi = \chi(a, b)$ is defined for an arc $(a, b) \in E$ by

$$\chi_v = \begin{cases} -1 & \text{if } v = a, \\ 1 & \text{if } v = b, \\ 0 & \text{otherwise.} \end{cases}$$

Denote by $A \in \mathbb{R}^{|E| \times |V|}$ the corresponding incidence matrix of G . Note that, for any subgraph G' of G , the corresponding subsystem of (1) defined by the arcs of G' is feasible if and only if G' is acyclic. Thus it follows that the simplices $\mathcal{S}(\mathcal{A})$ are in one-to-one correspondence with the *simple directed circuits* of G . By definition, an anti-simplex is a maximal subset of vectors not containing any simplex. Thus, the anti-simplices of \mathcal{A} correspond to the complements of the *minimal feedback arc sets* (i.e. minimal sets of arcs whose removal breaks every directed circuit in G).

Now, let us consider bodies and anti-bodies of \mathcal{A} (associated with the graph G). Fix a vertex $v \in V$ and consider the system of inequalities (2) together with the equation $x_v = 0$ (or equivalently, remove the v -th column of A and the v -th component of x). Then it is easy to see that the subsystem of (2) (together with $x_v = 0$) defined by the arcs of a subgraph G' of G is infeasible if and only if G' is strongly connected. In particular, the family of bodies $\mathcal{B}(\mathcal{A})$ is in one-to-one correspondence with the family of minimal strongly connected subgraphs of G , and the family of anti-bodies $\mathcal{B}(\mathcal{A})^{dc}$ is in one-to-one correspondence with the (complementary) family of minimal dicuts of G .

Given a directed graph $G = (V, E)$, it is known that all simple circuits of G can be listed with polynomial delay (see, e.g., [18]). It is also known [20] that all minimal feedback arc sets for a directed graph G can be listed with polynomial delay. Theorem 2 states that we can also list, in incremental polynomial time, all minimal strongly connected subgraphs of G , while Theorem 1 states that such a result is cannot hold for the family of minimal dicuts unless P=NP.

Thus, as a consequence of Theorem 1, we obtain the following negative result.

Corollary 1. *Given a set of vectors $\mathcal{A} \subseteq \mathbb{R}^n$, and a partial list $\mathcal{X} \subseteq \mathcal{B}(\mathcal{A})^{dc}$ of anti-bodies of S , it is NP-hard to determine if the given list is complete, i.e. $\mathcal{X} = \mathcal{B}(\mathcal{A})^{dc}$. Equivalently, given an infeasible system (2), and a partial list of maximal feasible subsystems of (2), it is NP-hard to determine if the given partial list is complete.*

We now turn to the enumeration of all bodies for \mathcal{A} . In contrast to Theorem 2, the general case of the enumeration problem for $\mathcal{B}(\mathcal{A})$ turns out to be at least as hard as the well-known *hypergraph transversal problem* [9], which is not known to be solvable in incremental polynomial time.

Proposition 1. *The problem of incrementally enumerating bodies, for a given set of $m + n$ points $\mathcal{A} \subseteq \mathbb{R}^n$, includes as a special case the problem of enumerating all minimal transversals for a given hypergraph \mathcal{H} with n hyperedges on m vertices.*

Proof. Given a hypergraph $\mathcal{H} = \{H_1, \dots, H_n\} \subseteq 2^{\{1, \dots, m\}}$, we define a set of points $\mathcal{A} \subseteq \mathbb{R}^n$, such that the bodies of \mathcal{A} are in one-to-one correspondence with the minimal transversals of \mathcal{H} . For $j = 1, \dots, n$, let \mathbf{e}_j be the j th unit vector, containing 1 in position j and 0 elsewhere. For $i = 1, \dots, m$, let $v^i \in \{0, 1\}^n$ be the vector with components $v_j^i = 1$ if $j \in H_i$ and $v_j^i = 0$ if $j \notin H_i$. Now

define $\mathcal{A} = \{-\mathbf{e}_1, \dots, -\mathbf{e}_m\} \cup \{v^1, \dots, v^m\}$. Let $X \in \mathcal{B}(\mathcal{A})$ be a body. If, for some $j \in \{1, \dots, n\}$, $-\mathbf{e}_j \notin X$, then $X \notin \mathcal{B}(\mathcal{A})$, because $X \subseteq \mathcal{A} \setminus \{-\mathbf{e}_j\} \subseteq \{x \in \mathbb{R}^n \mid x_j \geq 0\}$, and hence the convex hull of $\mathcal{A} \setminus \{-\mathbf{e}_j\}$ does not contain the origin as an interior point. We conclude therefore that X must contain the vectors $-\mathbf{e}_1, \dots, -\mathbf{e}_n$. Now it is easy to see that the set $X' = X \cap \{v^1, \dots, v^m\}$ is a minimal subset of vectors for which there exists, for each $j = 1, \dots, n$, a vector $v \in X'$ with $v_j = 1$, i.e. X' is a minimal transversal of \mathcal{H} . Conversely, let X be a minimal transversal of \mathcal{H} . Then X is a minimal set with the property that $\sum_{i \in X} v^i = y$, for some vector $y > 0$, and consequently the set of vectors $\{v^i : i \in X\} \cup \{-\mathbf{e}_1, \dots, -\mathbf{e}_n\}$ forms a body. \square

It should be mentioned that the best currently known algorithm for the hypergraph transversal problem runs in incremental *quasi-polynomial* time (see [10]). We also mention that the problem of generating simplices for a given set of points $\mathcal{A} \subseteq \mathbb{R}^n$ is equivalent with the well-known open problem of listing the vertices of a polytope given by its linear description:

Vertex Enumeration: Given an $m \times n$ real matrix $A \in \mathbb{R}^{m \times n}$ and an n -dimensional vector $b \in \mathbb{R}^n$ such that the polyhedron $P = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$ is bounded, enumerate all vertices of P .

If the polyhedron P is bounded, i.e. if it is a polytope, then the vertices of P are in one-to-one correspondence with the simplices of the point set \mathcal{A} whose elements are the columns of the augmented matrix $[A \mid -b]$. The complexity status of the vertex enumeration problem, and the transversal problem of enumerating anti-simplices, currently remains open. For the special case of vectors $A \subseteq \mathbb{R}^n$ in general position, we have $\mathcal{B}(\mathcal{A}) = \mathcal{S}(\mathcal{A})$, and consequently the problem of enumerating bodies of A turns into the problem of enumerating vertices of the polytope $\{x \in \mathbb{R}^n \mid Ax = 0, \mathbf{e}x = 1, x \geq \mathbf{0}\}$, each vertex of which is non-degenerate and has exactly $n + 1$ positive components. For such kinds of *simple* polytopes, there exist algorithms that generate all vertices with polynomial delay (see [2]). It is also worth mentioning that, for vectors in general position, the number of anti-bodies of \mathcal{A} is bounded by a polynomial in the number of bodies and n :

$$|\mathcal{B}(\mathcal{A})^{dc}| \leq (n+1)|\mathcal{B}(\mathcal{A})|, \quad (3)$$

see [15,19]. A polynomial inequality, similar to (3), for vectors not necessarily in general position, would imply that bodies, for any set of points $\mathcal{A} \subseteq \mathbb{R}^n$, could be generated in quasi-polynomial time (see, e.g., [6]). This follows from the fact that under the assumption that (3) holds, the problem of incrementally generating bodies reduces in polynomial time to the hypergraph transversal problem.

However, a polynomial bound similar to (3) does not hold in general as illustrated by the following example. Let $G = (V, E)$ be a directed graph on $k+2$ vertices consisting of two special vertices s, t , and k parallel directed (s, t) -path of length 2 each. Let $G\{s, t\}$ be the digraph obtained from G by adding, for each vertex $v \in V \setminus \{s, t\}$, two auxiliary vertices v', v'' and four auxiliary arcs $(t, v'), (v', v), (v, v''), (v'', s)$. Then it is not difficult to see that, for the set of incidence vectors $\mathcal{A} \subseteq \{-1, 0, 1\}^V$ corresponding to the arc-set of $G\{s, t\}$, the

number of bodies of \mathcal{A} is $|\mathcal{B}(\mathcal{A})| = k$, while the number of antibodies $|\mathcal{B}(\mathcal{A})^{dc}|$ exceeds 2^k .

Let us finally mention that, although the status of the problem of enumerating all maximal feasible subsystems of (1) is not known in general, the situation changes if we fix some set of inequalities, and ask for enumerating all its maximal extensions to a feasible subsystem. In fact, such a problem turns out to be NP-hard, even if we only fix non-negativity constraints.

Theorem 3. *Let $A \in \mathbb{R}^{m \times n}$ be an $m \times n$ matrix, $b \in \mathbb{R}^m$ be an m -dimensional vector, and assume that the system*

$$Ax \geq b, \quad x \in \mathbb{R}^n \tag{4}$$

has no solution $x \geq \mathbf{0}$. Let \mathcal{F} be the set of maximal subsystems of (4) for which there exists a non-negative solution x . Then given a partial list $\mathcal{X} \subseteq \mathcal{F}$, it is NP-hard to determine if the list is complete, i.e. if $\mathcal{X} = \mathcal{F}$, even if $b = (0, 0, \dots, 0, 1)$, and each entry in A is either, $-1, 1$, or 0 .

The proof of this theorem is omitted due to lack of space, but can be found in [5].

3 Proof of Theorem 1

Let us state first the following easy but useful characterization of minimal dicuts in a directed graph. For a directed graph $G = (V, E)$ and a subset $S \subseteq V$ of its vertices let us denote by $G[S]$ the subgraph of G induced by the vertex set S .

Lemma 1. *Given a strongly connected digraph $G = (V, E)$, an arc set $X \subseteq E$ is a minimal dicut if and only if there exist vertex sets $S, T, R \subseteq V$ such that*

- (D1) $S \neq \emptyset, T \neq \emptyset, S, T, R$ are pairwise disjoint, $S \cup T \cup R = V$,
- (D2) $G[S]$ and $G[T]$ are both strongly connected, and
- (D3) if $(a, b) \in E$ and $a \in S$ ($b \in T$) then $b \notin R$ ($a \notin R$), and
- (D4) $X = \{(a, b) \in E : a \in S, b \in T\}$ is the set of all arcs from S to T .

To prove the theorem, we use a polynomial transformation from the satisfiability problem. Let $\Phi = C_1 \wedge \dots \wedge C_m$ be a conjunctive normal form of m clauses and $2n$ literals $\{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$. In what follows, it is assumed without loss of generality that for each $i = 1, \dots, n$, both x_i and \bar{x}_i occur in Φ . We construct a strongly connected digraph $G = (V, E)$ with $|V| = m + 3n + 4$ vertices and $|E| = \sum_{i=1}^m |C_i| + m + 6n + 4$ arcs.

The vertices. The vertex set of G is defined as follows. There are m vertices C_1, \dots, C_m corresponding to the m clauses, $2n$ vertices $x_1, \bar{x}_1, \dots, x_n, \bar{x}_n$ corresponding to the $2n$ literals, $n + 1$ vertices p_0, p_1, \dots, p_n , and finally 3 other vertices z, u, y .

The arcs. There is an arc (z, C_j) from vertex z to every clause vertex C_j for $j = 1, \dots, m$, an arc (ℓ, y) from each literal $\ell \in \{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$ to vertex y ,

and an arc (C, ℓ) for each clause C and literal ℓ appearing in it. For $i = 1, \dots, n$, we also have arcs (p_{i-1}, x_i) , (p_{i-1}, \bar{x}_i) , (x_i, p_i) , and (\bar{x}_i, p_i) . Finally, we add the arcs (p_n, p_0) , (p_0, u) , (u, y) , and (y, z) .

Trivial dicuts. Let us call a minimal dicut *trivial* if it is the set of arcs leaving a single vertex $v \in V$, or the set of arcs entering a single vertex $v \in V$. Clearly, not all sets of arcs leaving or entering a vertex are minimal dicuts. However, the number of such minimal dicuts does not exceed twice the number of vertices, which is polynomial in n and m .

Non-trivial minimal dicuts. Let us now show that any non-trivial dicut yields a satisfying assignment for Φ and conversely, any satisfying assignment for Φ gives a non-trivial minimal dicut for G . This will prove Theorem 1.

Let $\sigma = (\ell_1, \ell_2, \dots, \ell_n)$ be the set of literals assigned the value *True* in a satisfying truth assignment for Φ . We define a minimal dicut X of G corresponding to σ . For this, we use the characterization of Lemma 1, i.e. give the corresponding sets S , T , and R :

$$T = \{p_0, \bar{\ell}_1, p_1, \bar{\ell}_2, \dots, p_{n-1}, \bar{\ell}_n, p_n\}, S = \{z, C_1, \dots, C_m, \ell_1, \dots, \ell_n, y\}, R = \{u\}. \quad (5)$$

To see the converse direction, let us consider a non-trivial minimal dicut $X \subseteq E$. We use Lemma 1 again to present a corresponding satisfying truth assignment for Φ . Since X is a minimal dicut, there exist sets S , T , and R satisfying conditions (D1)-(D4) of Lemma 1. We present the case analysis in the following steps:

(S0) None of the arcs (p_n, p_0) , (p_0, u) , (u, y) , (y, z) , and (z, C_j) for $j = 1, \dots, m$ can belong to X , since each of these arcs alone form a trivial minimal dicut.

(S1) We must have $|S| \geq 2$ and $|T| \geq 2$ by the non-triviality of X .

(S2) We must have $y \in S$, since otherwise all vertices from which y is reachable in $E \setminus X$ must belong to $R \cup T$ by (D3), implying by (S0) that $\{p_0, u, y\} \subseteq R \cup T$. Thus, $S \subseteq V \setminus \{p_0, u, y\}$ would follow, implying $|S| = 1$ in contradiction with (S1), since the vertex set $V \setminus \{p_0, u, y\}$ induces an acyclic subgraph of G .

(S3) Then, $\{y, z, C_1, \dots, C_m\} \subseteq S$ is implied by (S2), (D3) and (S0) (First, $z \notin R$ follows directly from (D3). Second $z \notin T$ follows from (S0) and (S2). Then (S0) implies that $C_j \in S$ for $j = 1, \dots, m$).

(S4) We must have $p_0 \in T$, since otherwise all vertices reachable from p_0 in $E \setminus X$ must belong to $S \cup R$ by (D3), implying by (S0) that $\{p_0, u, y\} \subseteq S \cup R$. Thus $T \subseteq V \setminus \{p_0, u, y\}$ would follow, implying $|T| = 1$ in contradiction with (S1), since the vertex set $V \setminus \{p_0, u, y\}$ induces an acyclic subgraph of G .

(S5) Then, $\{p_0, p_n\} \subseteq T$ is implied by (S4), (D3) and (S0).

(S6) The strong connectivity of $G[T]$ and (S5) then imply that there exist literals $\ell_i \in \{x_i, \bar{x}_i\}$ for $i = 1, \dots, n$ such that $\{p_0, \ell_1, p_1, \ell_2, \dots, p_{n-1}, \ell_n, p_n\} \subseteq T$.

(S7) The strong connectivity of $G[S]$ and (S3) then imply that there exist literals $\bar{\ell}_{i_j} \in S$ which belong to C_j for all $j = 1, \dots, m$. Furthermore, we are guaranteed by (S6) that $\ell_{i_j} \neq \bar{\ell}_{i_k}$ for all $k, j = 1, \dots, m$.

Now it is easy to see by (S7) and by the construction of the graph G that assigning $\bar{\ell}_{ij} \leftarrow \text{True}$, for all $j = 1, \dots, m$, yields a satisfying truth assignment for Φ . \square

4 Enumerating Minimal Strongly Connected Subgraphs

Let $G = (V, E)$ be a given strongly connected digraph and $\mathcal{F}_{sc} \subseteq 2^E$ the family of all minimal strongly connected subgraphs of G . We generate all elements of \mathcal{F}_{sc} by performing a traversal (for instance, breadth-first-search) of a directed “supergraph” $\mathcal{G} = (\mathcal{F}_{sc}, \mathcal{E})$ on vertex set \mathcal{F}_{sc} . Let $S \in \mathcal{F}_{sc}$ be a “vertex” of \mathcal{G} , then we define the neighborhood $\mathcal{E}^+(S) \subseteq \mathcal{F}_{sc}$ of the immediate successors of S in \mathcal{G} to consist of all minimal strongly connected subgraphs T of G which can be obtained from S by the following process:

1. Let $e = (a, b) \in S$ be an arc of G such that the graph $(V, E \setminus e)$ is strongly connected. Delete e from S .
2. Add a minimal set W of arcs from $E \setminus S$ to restore the strong connectivity of $(S \setminus e) \cup W$, i.e. the reachability of b from a .
3. Lexicographically delete some arcs Y from $S \setminus e$ to guarantee the minimality of $T = (S \setminus (Y \cup e)) \cup W$. (We assume in this step that we have fixed some order on the arcs of G).

Theorem 2 readily follows from the following lemma.

Lemma 2.

- (i) *The supergraph \mathcal{G} is strongly connected.*
- (ii) *For each vertex $S \in \mathcal{F}_{sc}$, the neighborhood of S can be generated with polynomial delay.*

Proof. (i) Let $S, S' \in \mathcal{F}_{sc}$ be two distinct vertices of \mathcal{G} . To show that \mathcal{G} contains an (S, S') -path, consider an arbitrary arc $e = (a, b) \in S \setminus S'$. Since $S \setminus \{e\} \cup S'$ is strongly connected, we can find a minimal set of arcs $W \subseteq S' \setminus S$ such that b is reachable from a in $S \setminus \{e\} \cup W$. Lexicographically minimizing the set of arcs $S \setminus \{e\} \cup W$ over $S \setminus e$, we obtain an element S'' in the neighborhood of S with a smaller difference $|S \setminus S'|$. This shows that \mathcal{G} is strongly connected and has diameter linear in n .

(ii) Let us start with the observation that for any two distinct minimal sets W and W' in Step 2, the neighbors resulting after Step 3 are distinct. Therefore, it suffices to show that all minimal arc sets W in Step 2 can be generated with polynomial delay.

For convenience, let us color the arcs in $S \setminus \{e\}$ black, and color the remaining arcs in $E \setminus S$ white. So we have to enumerate all minimal subsets W of white arcs such that b is reachable from a in $G(W) = (V, (S \setminus \{e\}) \cup W)$. Let us call such subsets of white arcs *minimal white (a, b) -paths*. The computation of these paths can be done by using a backtracking algorithm that performs depth first search on the following recursion tree \mathbf{T} (see [18] for general background on

backtracking algorithms). Each node (z, W_1, W_2) of the tree is identified with a vertex $z \in V$ and two disjoint subsets of white arcs W_1 and W_2 , such that W_1 is a minimal white (z, b) -path that can be extended to a minimal white (a, b) -path by adding some arcs from $W \setminus (W_1 \cup W_2)$. The root of the tree is $(b, \emptyset, \emptyset)$ and the leaves of the tree are those nodes (z, W_1, W_2) for which $a \in B(z)$, where $B(z)$ consists of all vertices $v \in V$ such that z can be reached from v in $S \setminus \{e\}$, that is by using only black arcs. As we shall see, the set W_1 for each leaf of T is a minimal white (a, b) -path, and each minimal white (a, b) -path will appear exactly once on a leaf of \mathbf{T} .

We now define the children of an internal node (z, W_1, W_2) , where $a \notin B(z)$. Let Z be the set of all white arcs from $W \setminus (W_1 \cup W_2)$ which enter $B(z)$. Pick an arc $e \in Z$. If the tail y of e is reachable from a in $G(W \setminus (W_1 \cup W_2 \cup Z))$, then $(y, W_1 \cup \{e\}, W_2 \cup (Z \setminus \{e\}))$ is a child of the node (z, W_1, W_2) in \mathbf{T} . It is easy to see that $W_1 \cup \{e\}$ is indeed a minimal white (y, b) -path: Let $e_1, e_2, \dots, e_k = e$ be the set of arcs added to $W_1 \cup \{e\}$, in that order, and let $(z^1, W_1^1, W_2^1), (z^2, W_1^2, W_2^2), \dots, (z^k, W_1^k, W_2^k)$ be the set of nodes from the root of tree \mathbf{T} to node $(y, W_1 \cup \{e\}, W_2 \cup (Z \setminus \{e\}))$. Then the set $W_1^k \setminus \{e_1\}$ does not contain any white path since it contains no arcs entering $B(b)$. More generally, for $i = 1, \dots, k$, the set $W_1^k \setminus \{e_i\}$ does not contain any white arc entering $B(b) \cup B(z^1) \cup \dots \cup B(z^{i-1})$ and hence it contains no white (y, b) -path. Note also that this construction guarantees that in addition to the minimality of the white (y, b) -path $W_1^k = W_1 \cup \{e\}$, it can be extended to a minimal white (a, b) path by adding some arcs from $(W \setminus (W_1 \cup W_2 \cup Z))$. Similar arguments also show that distinct leaves of \mathbf{T} yield distinct minimal white (a, b) -paths, and that all such distinct paths appear as distinct leaves in \mathbf{T} .

Note that the depth of the backtracking tree is at most $|V|$, and that the time spent at each node is polynomial in $|V|$ and $|E|$. This proves (ii). \square

As mentioned earlier, by performing a traversal on the nodes of the supergraph \mathcal{G} , we can generate the elements of \mathcal{F}_{sc} in incremental polynomial time. However, we cannot deduce from Lemma 2 that the set \mathcal{F}_{sc} can be generated with polynomial delay since the size of the neighborhood of a given vertex $S \in \mathcal{F}_{sc}$ may be exponentially large. Finally, we remark that the result of Theorem 2 can be extended to the case when we are given a fixed set of arcs, and would like to generate all minimal extensions that result in a strongly connected subgraph.

We conclude with the following observation. It is well known that the number of spanning trees (i.e., minimal edge sets ensuring the connectivity of the graph) for an undirected graph G can be computed in polynomial time (see, e.g., [4]). In contrast to this result, given a strongly connected digraph G with m arcs, it is NP-hard to approximate the size of \mathcal{F}_{sc} to within a factor of $2^{m^{1-\epsilon}}$, for any fixed $\epsilon > 0$. To see this, pick two vertices s and t in G and let $G\{s, t\}$ be the digraph obtained from G by the construction described in the end of Section 2. It is easy to see that any minimal strongly connected subgraph of $G\{s, t\}$ contains all the auxiliary arcs and some (s, t) -path in G . Hence there is a one-to-one correspondence between the set \mathcal{F}_{sc} for $G\{s, t\}$ and the set of directed

(s, t) -path for G^3 . Now the claim follows by using the amplification technique of [13], which replaces each arc of G by $(2m)^{1/\epsilon}$ consecutive pairs of parallel paths of length 2. It follows then that any approximation of the number of (s, t) -paths in the resulting graph G' to within an accuracy of $2^{(m')^{1-\epsilon}}$, where m' is the number of arcs in G' , can be used to compute the longest (s, t) -path in G , a problem that is known to be NP-hard.

A stronger inapproximability result for counting minimal dicuts is implied by the NP-hardness proof of Theorem 1: Unless P=NP, there is a constant $c > 0$, such that no polynomial-time algorithm can approximate the number of minimal dicuts of a given directed graph G to within a factor of 2^{cm} , where m is the number of arcs of G . This can be seen, for instance, as follows. Let $\Phi(x_1, \bar{x}_1, \dots, x_n, \bar{x}_n)$ be a CNF of k clauses on $2n$ literals. Replace Φ by

$$\Phi' = \Phi \wedge \bigwedge_{j=1}^s (y_j \vee z_j)(\bar{y}_j \vee \bar{z}_j),$$

where y_1, \dots, y_s and z_1, \dots, z_s are new variables. This way we obtain a new CNF Φ' of $k + 2s$ clauses on $2n + 2s$ literals such that Φ has a satisfying assignment if and only if Φ' has at least 2^s satisfying assignments. Let G be the digraph with $O(k + n + s)$ arcs constructed for Φ' in the proof of Theorem 1. Now the result follows from the fact that it is NP-hard to determine whether G has $O(k + n + s)$ or more than 2^s minimal dicuts.

Acknowledgements

We thank Marc Pfetsch for helpful discussions.

References

1. U. Abel and R. Bicker, Determination of all cutsets between a node pair in an undirected graph, *IEEE Transactions on Reliability* **31** (1986) pp. 167-171.
2. D. Avis, K. Fukuda, A Pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra, *Symp. Comput. Geom.* 1991, pp. 98-104.
3. V.K. Bansal, K.B. Misra and M.P. Jain, Minimal pathset and minimal cutset using search technique, *Microelectr. Reliability* **22** (1982) pp. 1067-1075.
4. B. Bollobas, Graph Theory: An introductory course, Springer Verlag, 1979.
5. E. Boros, K. Elbassioni, V. Gurvich and L. Khachiyan, On enumerating minimal dicuts and strongly connected subgraphs, DIMACS Technical Report 2003-35, Rutgers University, <http://dimacs.rutgers.edu/TechnicalReports/2003.html>.
6. E. Boros, V. Gurvich, L. Khachiyan and K. Makino, Dual-bounded generating problems: partial and multiple transversals of a hypergraph, *SIAM J. Comput.*, **30** (2001), pp. 2036–2050.

³ In particular, this shows that maximizing the number of arcs in a minimal strongly connected subgraph of G is NP-hard. Minimizing the number of arcs in such a subgraph is also known to be NP-hard [11]

7. C. J. Colburn, The Combinatorics of network reliability. Oxford Univ. Press, 1987.
8. N.D. Curet, J. DeVinney, and M.E. Gaston, An efficient network flow code for finding all minimum cost s - t cutsets, *Comp. and Oper. Res.* **29** (2002), pp. 205-219.
9. T. Eiter and G. Gottlob, Identifying the minimal transversals of a hypergraph and related problems, *SIAM J. Comput.*, 24 (1995), pp. 1278-1304.
10. M. L. Fredman and L. Khachiyan (1996), On the complexity of dualization of monotone disjunctive normal forms, *J. Algorithms*, **21**, pp. 618-628.
11. M. R. Garey and D. S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W.H. Freeman and Co., 1979.
12. D. Gusfield and D. Naor, Extracting maximum information about sets of minimum cuts, *Algorithmica* **10** (1993) pp. 64-89.
13. M.R. Jerrum, L.G. Valiant and V.V. Vazirani, Random generation of combinatorial structures from a uniform distribution, *Theor. Comp. Sc.*, **43** (1986) pp. 169-188.
14. E. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan, Generating all maximal independent sets: NP-hardness and polynomial-time algorithms, *SIAM Journal on Computing*, 9 (1980) pp. 558-565.
15. M. Pfetsch, The Maximum feasible Subsystem Problem and Vertex-Facet Incidences of Polyhedra, Dissertation, TU Berlin, 2002.
16. J. S. Provan and M. O. Ball, Computing network reliability in time polynomial in the number of cuts, *Operations Research* **32** (1984) pp. 516-526.
17. J. S. Provan and D. R. Shier, A paradigm for listing (s, t) cuts in graphs, *Algorithmica* **15**, (1996), pp. 351-372.
18. R. C. Read and R. E. Tarjan, Bounds on backtrack algorithms for listing cycles, paths, and spanning trees, *Networks*, 5 (1975) pp. 237-252.
19. J. Ryan, IIS-Hypergraphs, *SIAM J. Discrete Math.*, 9(4) 1996, pp. 643-653.
20. B. Schwikowski, E. Speckenmeyer, On enumerating all minimal solutions of feedback problems, *Discrete Applied Mathematics*, **117** (2002), pp. 253-265.

Semi-continuous Cuts for Mixed-Integer Programming^{*}

I.R. de Farias Jr.

Department of Industrial Engineering
University at Buffalo, SUNY
defarias@buffalo.edu

Abstract. We study the convex hull of the feasible set of the *semi-continuous knapsack problem*, in which the variables belong to the union of two intervals. Besides being important in its own right, the semi-continuous knapsack problem is a relaxation of general mixed-integer programming. We show how strong inequalities that are valid for the *semi-continuous knapsack polyhedron* can be derived and used as cuts in a branch-and-cut scheme for mixed-integer programming and problems with semi-continuous variables. We present computational results that demonstrate the effectiveness of these inequalities, which we call collectively *semi-continuous cuts*. Our computational experience also shows that dealing with semi-continuous constraints directly in the branch-and-cut algorithm through a specialized branching scheme and semi-continuous cuts is considerably more practical than the “textbook” approach of modeling semi-continuous constraints through the introduction of auxiliary binary variables in the model.

Keywords: mixed-integer programming, semi-continuous variables, disjunctive programming, polyhedral combinatorics, branch-and-cut

1 Introduction

Let n be a positive integer, $N = \{1, \dots, n\}$, N^+ , N_∞^+ , and N^- three disjoint subsets of N with $N^+ \cup N_\infty^+ \cup N^- = N$, and $a_j > 0 \forall j \in N^+ \cup N_\infty^+$ and $a_j < 0 \forall j \in N^-$. We study the inequality description of $P = \text{clconv}(S)$, the closure of the convex hull of S , where $S = \{x \in \mathbb{R}^n : x \text{ satisfies (1) -- (3)}\}$, and

$$\sum_{j \in N} a_j x_j \leq b, \quad (1)$$

$$x_j \in [0, p_j] \cup [l_j, u_j] \quad \forall j \in N^+, \quad (2)$$

$$x_j \in [0, p_j] \cup [l_j, \infty) \quad \forall j \in N_\infty^+ \cup N^-. \quad (3)$$

* Partially supported by NSF grants DMI-0100020 and DMI-0121495. This paper is an extended abstract of [7].

When $p_j < l_j$, we call x_j a *semi-continuous variable*, and the corresponding constraint (2) or (3) a *semi-continuous constraint*. Note that our definition of semi-continuous variable is more general than the usual one (Beale [2]), in which $p_j = 0$. When $p_j \geq l_j$, we say that x_j is a continuous variable. The set P is the *semi-continuous knapsack polyhedron*. An optimization problem whose constraints are (1), (2), and (3) is a *semi-continuous knapsack problem* (SCKP).

Semi-continuous constraints appear in general mixed-integer programming (MIP), since $x_j \in \mathbb{Z} \cap [0, u_j] \Rightarrow x_j \in [0, p_j] \cup [p_j + 1, u_j]$, where p_j is an integer. They also appear more explicitly in other applications, such as production scheduling (Beale [2]) and portfolio optimization (Bienstock [4], Perold [18]).

The “textbook” approach to dealing with the semi-continuous constraints (2) is to introduce a binary variable y_j in the model for each semi-continuous variable x_j , $j \in N^+$, and the constraints

$$x_j \geq l_j y_j \quad (4)$$

and

$$x_j \leq p_j + (u_j - p_j)y_j, \quad (5)$$

see for example Nemhauser and Wolsey [17]. This approach has the potential disadvantage of considerably increasing: (i) the size of the problem, since the inclusion of the auxiliary binary variables doubles the number of variables, and the inclusion of (4) and (5) adds $2n$ constraints to the model; (ii) the amount of branching, since it may happen that in an optimal solution x to the LP relaxation, x satisfies (2) but $y_j \in (0, 1)$ for some $j \in N^+$, in which case branching on y_j may be performed to enforce integrality; (iii) degeneracy of the LP relaxation, due to (4) and (5); see de Farias et al. [9]. Note that to use the same device to model (3) it is necessary to assign *big-M* values to u_j in (5). Among other issues, the introduction of big- M constants may cause severe numerical difficulties. For example, in a basic solution to the LP relaxation, many times some of the constraints (5) are satisfied at equality. If $x_j \in (p_j, l_j)$ but it is much smaller than the big- M constant assigned to it, y_j will be very small, and the solver may consider it to be 0. As a result, the solver may consider the LP relaxation solution feasible (i.e. such that $y_j \in \{0, 1\} \forall j \in N$) and fathom the node. The common end result is that the solver gives as optimal solution one that does not satisfy all semi-continuous constraints.

An alternative to the “textbook” approach that dispenses with the introduction of the auxiliary binary variables y_j and the constraints (4) and (5) in the model was suggested by Beale [2]. It consists of branching directly on the semi-continuous variables x_j by adding the bound $x_j \leq p_j$ in one branch and $x_j \geq l_j$ in the other branch, thus enforcing the semi-continuous constraints (2) and (3) algorithmically, directly in the branch-and-bound scheme. Note that, this way, all the issues explained in the previous paragraph are immediately resolved.

We explore Beale’s approach further by deriving strong cuts in the x -space valid for P and using them in a *branch-and-cut scheme without auxiliary binary variables* (de Farias et al. [9]) for solving problems that contain semi-continuous

constraints. Branch-and-cut without auxiliary binary variables has been successfully used in the study of other combinatorial constraints, including complementarity (de Farias et al. [10], Ibaraki et al. [14]), cardinality (Bienstock [4], de Farias and Nemhauser [11], Laundy [15]), and special ordered sets (Beale and Tomlin [3], de Farias et al. [8]).

The main tool used in this paper to study the inequality representation of semi-continuous knapsack polyhedra in the x -space is lifting, see Louveaux and Wolsey [16] for the basic theory of lifting and recent developments. As part of our lifting approach, we generalize the concepts of cover and cover inequality (Balas [1], Hammer et al. [13], Wolsey [19]), which have proven to be of great use in 0-1 programming, for semi-continuous constraints. We also show how to lift our cover inequalities to derive strong cuts to be used in branch-and-cut. We call our cuts collectively *semi-continuous cuts*.

Even though our theoretical results are general, we test their practical value specifically on problems with integer constraints, semi-continuous constraints, and the constraints

$$x_j \in [0, r_j] \cup \{s_j\} \cup [t_j, u_j], j \in N,$$

which we call *three-term semi-continuous*. They arise, for example, in financial applications, see Perold [18]. We show through our computational results that semi-continuous cuts can be effective for general MIP, and problems with semi-continuous and three-term semi-continuous constraints. Our computational results also demonstrate that branch-and-cut without auxiliary binary variables is superior to the “textbook” MIP approach for problems with semi-continuous and three-term semi-continuous constraints.

Compared to the 0-1 case, polyhedral studies for general MIP are scarce in the literature. For a review on the use of polyhedral methods for MIP, see the recent study by Louveaux and Wolsey [16], which contains polyhedral results for constraints (4) and (5). We are not aware of any polyhedral study for semi-continuous or three-term semi-continuous constraints.

In Sect. 2 we introduce assumptions and we present a few simple results on P . In Sect. 3 we present lifting results that hold for the nontrivial inequalities for P . In Sect. 4 we extend the concepts of cover and cover inequality of 0-1 programming to our case and we derive lifting results for cover inequalities. In Sect. 5 we present results of our computational experience on the effectiveness of semi-continuous cuts and branch-and-cut without auxiliary binary variables.

2 The Semi-continuous Knapsack Polyhedron

In this section we introduce assumptions and we present a few simple results about P . We establish the *trivial* facet-defining inequalities and when they suffice to describe P . We present a *nontrivial* facet-defining inequality that dominates (1) and under certain conditions is the only nontrivial facet-defining inequality for P . Finally, we present a few relations that hold among the coefficients of the variables in a nontrivial inequality.

We will assume throughout the paper that: (a) $n \geq 2$; (b) $p_j \geq 0$ and $l_j > 0 \forall j \in N$, and $u_j \geq l_j \forall j \in N^+$; (c) when $N^- = \emptyset$, $N_\infty^+ = \emptyset$ and $a_j u_j \leq b \forall j \in N^+$; (d) when $N_\infty^+ = \emptyset$, $\sum_{j \in N^+} a_j u_j > b$. When Assumption (a) does not hold, the problem is trivial. So, there is no loss of generality in Assumption (a). In addition, with Assumption (a) it is possible to simplify the presentation of several results that would otherwise have to consider separately the case $n = 1$. If $u_j = 0$ for some $j \in N^+$, x_j can be eliminated from the problem. Therefore, there is no loss of generality in Assumption (b). Assumption (b) implies that $p_j > 0$ whenever x_j is continuous. When $N^- = \emptyset$, x_j is bounded $\forall j \in N$. In addition, for $j \in N^+$, it is possible, in this case, to scale x_j so that $a_j u_j \leq b$, unless $b = 0$ or $x_j \in \{0\} \cup [l_j, u_j]$ with $a_j l_j > b$. In the first case the problem is trivial, and in the second case x_j can be eliminated from the problem. Thus, there is no loss of generality in Assumption (c). Assumption (c) implies that when $N^- = \emptyset$, $b > 0$. Finally, if $N_\infty^+ = \emptyset$, (1) is redundant unless $\sum_{j \in N^+} a_j u_j > b$. This means that there is no loss of generality in Assumption (d) either. Assumption (d) implies that when $N^+ \cup N_\infty^+ = \emptyset$, $b < 0$.

As a result of Assumptions (a)–(d), Propositions 1–3 follow. \square

Proposition 1. *P* is full-dimensional. \square

Proposition 2. *The inequality*

$$x_j \geq 0 \quad (6)$$

is facet-defining $\forall j \in N^+ \cup N_\infty^+$. For $j \in N^-$, (6) is facet-defining iff either: (i) $|N^-| > 1$; or (ii) $b > 0$ and $\forall k \in N^+ \cup N_\infty^+$ either $p_k > 0$ or $a_k l_k \leq b$. \square

Proposition 3. When $N^- \neq \emptyset$,

$$x_j \leq u_j \quad (7)$$

is facet-defining $\forall j \in N^+$. When $N^- = \emptyset$, (7) is facet-defining iff $a_j u_j < b$ and $\forall k \in N^+ - \{j\}$ either $p_k > 0$ or $a_j u_j + a_k l_k \leq b$. \square

Example 1. Let $S = \{x \in \mathbb{R}_+^3 : 2x_1 + 3x_2 \leq 4 + x_3, x_1 \in [0, 1] \cup [2, 3], x_2 \in \{0\} \cup [3, \infty)\}$. Then, $x_1 \geq 0$, $x_1 \leq 3$, $x_2 \geq 0$ are facet-defining for *P*. On the other hand, $x_3 \geq x_2 \forall x \in S$, and therefore $x_3 \geq 0$ is not facet-defining. \square

Unlike inequalities (6) and (7), there do not seem to exist simple necessary and sufficient conditions to determine when (1) is facet-defining. We now present an inequality that, when $N = N^-$, is valid for *P*, is at least as strong as (1) (and possibly stronger), and under an additional condition gives, together with (6), a full inequality description for *P*.

Proposition 4. Suppose that $N = N^-$, and let $N_0^- = \{j \in N^- : p_j = 0\}$. Then,

$$\sum_{j \in N_0^-} \frac{a_j}{\min\{b, a_j l_j\}} x_j + \frac{1}{b} \sum_{j \in N^- - N_0^-} a_j x_j \geq 1 \quad (8)$$

is valid for P . If $N^- = N_0^-$ or $\exists k \in N^- - N_0^-$ such that x_k is continuous, then (8) is facet-defining. If $N^- = N_0^-$, then $P = \{x \in \mathbb{R}_+^n : x \text{ satisfies (8)}\}$.

Example 2. Let $S = \{x \in \mathbb{R}_+^5 : 2x_1 + 3x_2 + 3x_3 + x_4 \geq 4 + x_5, x_1 \in \{0\} \cup [1, \infty), x_2 \in \{0\} \cup [2, \infty), x_3 \in [0, 1] \cup [2, \infty), x_5 \in [0, 1] \cup \{2\}\}$. Then, $P \cap \{x \in \mathbb{R}^5 : x_3 = x_4 = x_5 = 0\} = \{x \in \mathbb{R}_+^5 : x_1 + x_2 \geq 2, x_3 = x_4 = x_5 = 0\}$. \square

Let $PR = \{x \in \mathbb{R}_+^n : x \text{ satisfies (1), and (7) } \forall j \in N^+\}$, the feasible set of the LP relaxation of SCKP. We now give a necessary and sufficient condition for $P = PR$.

Proposition 5. $P = PR$ iff $\forall T \subseteq N^+, i \in N^+ \cup N_\infty^+ - T$, and $k \in N^-$, the following two conditions are satisfied: (i) $\sum_{j \in T} a_j u_j + a_i p_i \geq b$ or $\sum_{j \in T} a_j u_j + a_i l_i \leq b$; and (ii) $\sum_{j \in T} a_j u_j + a_k p_k \leq b$ or $\sum_{j \in T} a_j u_j + a_k l_k \geq b$. \square

Inequalities (1), (6), (7), and the inequalities dominated by one of them are called *trivial*. For the remainder of the paper we will study the *nontrivial* inequalities for P . We will denote a generic nontrivial inequality valid for P as

$$\sum_{j \in N} \alpha_j x_j \leq \beta. \quad (9)$$

Note that $\alpha_j \leq 0 \forall j \in N^-$, $\beta > 0$ whenever $N^- = \emptyset$, and $\beta < 0$ whenever $N^+ \cup N_\infty^+ = \emptyset$. In addition, (9) remains valid if we replace α_j with $(\alpha_j)^+$ $\forall j \in N^+ \cup N_\infty^+$, and we will then assume that $\alpha_j \geq 0 \forall j \in N^+ \cup N_\infty^+$.

Proposition 6. Let $k \in N^-$. If: (i) $i \in N_\infty^+$ and (9) is satisfied at equality for at least one point \tilde{x} of S , then $\frac{\alpha_i}{a_i} \leq \frac{\alpha_k}{a_k}$; (ii) $i \in N^+$, x_k is continuous, and (9) is satisfied at equality for at least one point \tilde{x} of S with $\tilde{x}_i < u_i$, then $\frac{\alpha_i}{a_i} \leq \frac{\alpha_k}{a_k}$, or in case $\tilde{x}_i > 0$, then $\frac{\alpha_i}{a_i} \geq \frac{\alpha_k}{a_k}$; (iii) $i \in N^-$, x_k is continuous, and (9) is satisfied at equality for at least one point \tilde{x} of S with $\tilde{x}_i > 0$, then $\frac{\alpha_i}{a_i} \leq \frac{\alpha_k}{a_k}$. \square

Proposition 7. Suppose that (9) is facet-defining. If $i \in N_\infty^+$ and x_i is continuous, then $\alpha_i = 0$. \square

From statement (iii) of Proposition 6 it follows that if (9) is facet-defining, and x_i and x_k , $i, k \in N^-$, are continuous, $\frac{\alpha_i}{a_i} = \frac{\alpha_k}{a_k}$. Because of this and Proposition 7 we will continue our polyhedral study under the following two assumptions: (e) there is at most one continuous variable x_i with $i \in N^-$; (f) $p_j < l_j \forall j \in N_\infty^+$.

3 Lifting

In this section we present lifting results that hold for the nontrivial inequalities for P . We show that in some cases it is easy to obtain the exact lifting coefficients of several variables. We then show how all the lifting coefficients can be calculated approximately in time $O(n^2)$. For a recent review on the lifting technique see Louveaux and Wolsey [16].

Let $\tilde{x} \in P$. We will henceforth denote a generic nontrivial valid inequality for $P_T = P \cap \{x \in \mathbb{R}^n : x_j = \tilde{x}_j \forall j \in N - T\}$ as

$$\sum_{j \in T} \alpha_j x_j \leq \beta. \quad (10)$$

As we establish in Propositions 8 and 9, in some cases it is easy to obtain the exact lifting coefficients of several variables. In Proposition 8 we establish that when $\tilde{x}_i = 0$, $i \in (N^+ \cup N_\infty^+) - T$, and $p_i > 0$, the lifting coefficient of x_i is equal to 0.

Proposition 8. *Let $i \in (N^+ \cup N_\infty^+) - T$, $\tilde{x}_i = 0$, and suppose that (10) defines a facet of P_T . If $p_i > 0$, the lifting coefficient of x_i is 0.* \square

Example 2 (Continued). The lifting coefficient of x_5 is 0, regardless of the lifting order. Therefore $P \cap \{x \in \mathbb{R}^5 : x_3 = x_4 = 0\} = \{x \in \mathbb{R}_+^5 : x_1 + x_2 \geq 2, x_3 = x_4 = 0\}$. \square

We now establish that when (10) contains a continuous variable x_k with $k \in N^-$, it is easy to obtain the lifting coefficient of a variable x_i with $i \in N^-$ and $p_i > 0$ that is fixed at 0, or with $i \in N^+$ and $l_i < u_i$ that is fixed at u_i .

Proposition 9. *Let x_k , $k \in T^-$, be a continuous variable, and suppose that (10) defines a facet of P_T . If: (i) $i \in N^- - T$, $p_i > 0$, and $\tilde{x}_i = 0$; or (ii) $i \in N^+ - T$, $l_i < u_i$, and $\tilde{x}_i = u_i$; the lifting coefficient of x_i is $\alpha_i = \frac{a_i}{a_k} \alpha_k$.* \square

Example 2 (Continued). The lifting coefficient of x_4 is the optimal value of

$$\max \left\{ \frac{2 - x_1 - x_2}{x_4} : x \in S, x_4 > 0, x_3 = 0 \right\}. \quad (11)$$

Let x^* be an optimal solution to (11). Because $x_2 > 0 \Rightarrow x_2 \geq 2$, $x_2^* = 0$. In addition, it is clear that $2x_1^* + x_4^* = 4$. Thus,

$$\max \left\{ \frac{2 - x_1 - x_2}{x_4} : x \in S, x_4 > 0, x_3 = 0 \right\} = \frac{2 - x_1^*}{4 - 2x_1^*} = \frac{1}{2}.$$

So, $2x_1 + 2x_2 + x_4 \geq 4$ defines a facet of $P \cap \{x \in \mathbb{R}^5 : x_3 = 0\}$. From Proposition 9, if we now lift x_3 , the lifting coefficient is 3, and thus $2x_1 + 2x_2 + 3x_3 + x_4 \geq 4$ defines a facet of P . \square

In 0-1 programming the lifting problem is a linear 0-1 knapsack problem. In practice, it is common, in this case, to solve the lifting problem approximately by solving its LP relaxation and rounding the resulting optimal value; see Gu et al. [12], where an extensive computational study on 0-1 lifting is presented that shows, among other things, that it is more practical to use the LP relaxation approximation to compute the lifting coefficients than to use dynamic programming to compute them exactly. In the case of semi-continuous variables, however, the lifting problem is a fractional knapsack problem on semi-continuous variables. We now show how to solve the LP relaxation of the lifting problem for this case to obtain approximate values for the lifting coefficients of x_j , $j \in N - T$, in (10). As in 0-1 programming, the procedure gives all lifting coefficients approximately in time $O(n^2)$. We will discuss specifically the case where the next variable to be lifted is x_k , $k \in N^+$, $p_k = 0$, $p_k < l_k < u_k$, and $\tilde{x}_k = 0$. The other cases can be treated in a similar way.

Let $k \in N^+ - T$. Suppose that $p_k = 0$, $p_k < l_k < u_k$, $\tilde{x}_k = 0$, and that we lift (10) next with respect to x_k . The approximate lifting coefficient α_k of x_k is given by the optimal value of the LP relaxation of the lifting problem, i.e.

$$\alpha_k = \min \left\{ \frac{\beta - \sum_{j \in T} \alpha_j x_j}{x_k} : x \in \bar{S}_k \right\}, \quad (12)$$

where $\bar{S}_k = \{x \in \mathbb{R}_+^n : \sum_{j \in T} a_j x_j + a_k x_k \leq b - \sum_{j \in N-T} a_j \tilde{x}_j, x_j \leq u_j \forall j \in N^+, x_j = \tilde{x}_j \forall j \in N-(T \cup \{k\}), \text{ and } x_k \geq l_k\}$. We assume that $\bar{S}_k \neq \emptyset$, otherwise x_k cannot be the next lifted variable. Rather than solving (12), we solve

$$\max \left\{ \sum_{j \in T} \alpha_j x_j + \tilde{\alpha}_k x_k : x \in \bar{S}_k \right\} = \beta, \quad (13)$$

where the variables are $\tilde{\alpha}_k$ and x_j , $j \in N$. Problem (13) is to find a value $\tilde{\alpha}_k^*$ for the variable $\tilde{\alpha}_k$ such that $\forall x \in \bar{S}_k \sum_{j \in T} \alpha_j x_j + \tilde{\alpha}_k^* x_k \leq \beta$, and values x_j^* for the variables x_j , $j \in N$, such that $x^* \in \bar{S}_k$ and $\sum_{j \in T} \alpha_j x_j^* + \tilde{\alpha}_k^* x_k^* = \beta$. We now show that problems (12) and (13) are equivalent.

Proposition 10. *Let α_k be the optimal value of (12) and $(\tilde{\alpha}_k^*, x^*)$ be a solution to (13). Then, $\alpha_k = \tilde{\alpha}_k^*$.* \square

Suppose WLOG that $T = \{1, \dots, t\}$ and

$$\frac{\alpha_1}{a_1} \geq \dots \geq \frac{\alpha_t}{a_t}. \quad (14)$$

We now show how to solve (13) in linear time, once $\frac{\alpha_1}{a_1}, \dots, \frac{\alpha_t}{a_t}$ have been sorted as in (14). We first note that the maximization problem in (13) is never unbounded. Now, if we know whether

$$\frac{\alpha_k}{a_k} \geq \frac{\alpha_1}{a_1}, \quad (15)$$

or

$$\frac{\alpha_j}{a_j} \geq \frac{\alpha_k}{a_k} \geq \frac{\alpha_{j+1}}{a_{j+1}} \quad (16)$$

for some $j \in \{1, \dots, t-1\}$, or else

$$\frac{\alpha_t}{a_t} \geq \frac{\alpha_k}{a_k}, \quad (17)$$

then it is possible to find an optimal solution x^* to the maximization problem in (13), even if we do not know the value of α_k . But then,

$$\alpha_k = \frac{\beta - \sum_{j \in T} \alpha_j x_j^*}{x_k^*}. \quad (18)$$

We determine α_k by calculating the values that result from (18) under the cases (15)–(17). Specifically, let α_k^0 be the value of α_k resulting from case (15), α_k^j , $j \in \{1, \dots, t-1\}$, the values of α_k resulting from case (16), and α_k^t the value of α_k resulting from case (17). Then, $\alpha_k = (\min\{\alpha_k^0, \dots, \alpha_k^t\})^+$.

By calculating $\alpha_k^t, \dots, \alpha_k^0$ in this order, it is possible to calculate all of them in time $O(n)$. Also, by maintaining a list of the variable indices sorted as in (14) every time a new lifting coefficient is calculated, it will not be necessary to sort the indices, except for the initial inequality, before any variable is lifted. Therefore, all lifting coefficients can be calculated approximately in time $O(n^2)$.

Example 3. Let $S = \{x \in \mathbb{Z}^3 : 4x_1 + 3x_2 + 3x_3 \leq 16, x_1 \in [0, 1] \cup \{2\}, x_2 \in [0, 2] \cup [3, 4], x_3 \in \{0\} \cup [1, 3]\}$. As we will show in Sect. 4, $2x_1 + x_2 \leq 6$ defines a facet of $\text{conv}(S) \cap \{x \in \mathbb{R}^3 : x_3 = 0\}$. We now lift x_3 approximately. The approximate lifting problem is $\max\{2x_1 + x_2 + \alpha_3 x_3 : 4x_1 + 3x_2 + 3x_3 \leq 16, x_1 \in [0, 2], x_2 \in [0, 4], x_3 \in [1, 3]\} = 6$. If $\frac{\alpha_3}{3} \leq \frac{1}{3}$, then $x_1^{(2)} = 2, x_2^{(2)} = \frac{5}{3}, x_3^{(2)} = 1$ is an optimal solution, and $\alpha_3^2 = \frac{1}{3}$. If $\frac{1}{3} \leq \frac{\alpha_3}{3} \leq \frac{1}{2}$, then $x_1^{(1)} = 2, x_2^{(1)} = 0, x_3^{(1)} = \frac{8}{3}$ is an optimal solution, and $\alpha_3^1 = \frac{8}{3}$. If $\frac{1}{2} \leq \frac{\alpha_3}{3}$, then $x_1^{(0)} = \frac{7}{4}, x_2^{(0)} = 0, x_3^{(0)} = 3$ is an optimal solution, and $\alpha_3^0 = \frac{5}{6}$. Thus, the approximate lifting coefficient is $\alpha_3 = \frac{1}{3}$, which coincides with the exact lifting coefficient. \square

4 Cover Inequalities

In this section we extend the concepts of cover and cover inequality (Balas [1], Hammer et al. [13], Wolsey [19]), commonly used in 0-1 programming, to our case. We show that when the cover is *simple*, the cover inequality is the only nontrivial facet-defining inequality of the semi-continuous knapsack polyhedron obtained by fixing at 0 all variables not indexed by the cover. We then give the value of the lifting coefficient of a continuous variable x_k , $k \in N^-$, that is fixed at 0, when the cover inequality is lifted with respect to it first. Finally, we show that for mixed-integer programming, our cover inequality is sometimes stronger than the Gomory mixed-integer cut and the cover inequality defined in [6].

We now define cover and simple cover for semi-continuous knapsack polyhedra.

Definition 11. Let $C \subseteq N^+$ with $p_j < l_j \forall j \in C$. We say that C is a cover if $\sum_{j \in C} a_j l_j > b$. If in addition $\sum_{j \in C - \{i\}} a_j u_j + a_i p_i \leq b \forall i \in C$, we say that the cover is simple. \square

Note that our definitions of cover and simple cover coincide with the definitions of cover and minimal cover of 0-1 programming, where $p_j = 0$ and $l_j = u_j = 1 \forall j \in N$.

We now introduce cover inequalities and show that they are the only non-trivial facet-defining inequalities for $P^0 = P \cap \{x \in \mathbb{R}^n : x_j = 0 \forall j \in N - C\}$ when the cover is simple.

Proposition 12. Let C be a cover. Then,

$$\sum_{j \in C} \frac{u_j - x_j}{u_j - p_j} \geq 1 \quad (19)$$

is valid for P . If C is a simple cover, then (19) is facet-defining for P^0 , and $P^0 = \{x \in \mathbb{R}^n : x \text{ satisfies (6), (7), and (19)}\}$. \square

When $p_j = 0 \forall j \in C$, i.e. for the “usual” semi-continuous variables, (19) becomes $\sum_{j \in C} \frac{x_j}{u_j} \leq |C| - 1$. In particular, for the case of 0-1 variables, (19) becomes $\sum_{j \in C} x_j \leq |C| - 1$.

Example 4. Let $S = \{x \in \mathbb{R}_+^5 : 2x_1 + 3x_2 + 4x_3 + x_4 + x_5 \leq 33, x_1 \in \{0\} \cup [2, 3], x_2 \in [0, 1] \cup [3, 4], x_3 \in [0, 2] \cup [4, 5], x_4 \in \{0\} \cup [1, 2], x_5 \in [0, 1] \cup \{2\}\}$. Note that $\sum_{j \in N} a_j l_j = 32 < 33$, and therefore N is not a cover. However, by fixing $x_1 = 3, x_2 = 4$, and $x_4 = x_5 = 0$, $\{3\}$ becomes a simple cover, and $x_3 \leq 2$ is valid and facet-defining for $P \cap \{x \in \mathbb{R}^5 : x_1 = 3, x_2 = 4, x_4 = x_5 = 0\}$. \square

We now give the lifting coefficient of a continuous variable $x_k, k \in N^-$, that is fixed at 0, when the cover inequality is lifted with respect to it first.

Proposition 13. Let C be a simple cover and $x_k, k \in N^-$, a continuous variable that is fixed at 0. Then,

$$\sum_{j \in C} \frac{u_j - x_j}{u_j - p_j} - \frac{a_k}{\sum_{j \in C} a_j l_j - b} \left(1 - \sum_{j \in C} \frac{u_j - l_j}{u_j - p_j} \right) x_k \geq 1 \quad (20)$$

is facet-defining for $P_{C \cup \{k\}}$. \square

We now show in Example 5 that our cover inequality can be stronger than the Gomory mixed-integer cut and the cover inequality defined in [6]. The data for the example is taken from [6], Example 2.2.

Example 5. Let $S_1 = \{(x, s) \in \mathbb{Z}_+^2 \times \mathbb{R}_+ : 4x_1 + 3x_2 \leq 16 + s, x_1 \leq 2, x_2 \leq 3\}$ and $S_2 = \{(x, s) \in \mathbb{Z}_+^2 \times \mathbb{R}_+ : 4x_1 + 3x_2 \leq 16 + s, x_1 \leq 2, x_2 \leq 4\}$. The point $(x^*, s^*) = (2, \frac{8}{3}, 0)$ is a fractional vertex of the LP relaxation for both sets. (It is the only fractional vertex in the case of S_2 .)

Consider S_1 . The Gomory mixed-integer cut for the basis of (x^*, s^*) is $2x_1 + x_2 \leq 6 + s$. Inequality (20) for $x_1 \in [0, 1] \cup \{2\}$ and $x_2 \in [0, 2] \cup \{3\}$ is $x_1 + x_2 \leq 4 + s$, which is facet-defining and stronger than the Gomory inequality, since $(2x_1 + x_2 \leq 6 + s) = (x_1 + x_2 \leq 4 + s) + (x_1 \leq 2)$. Note that in this case the semi-continuous inequality coincides with the cover inequality of Ceria et al. [6].

Consider now S_2 . In [6] the following cover inequality is proposed $x_1 + x_2 \leq 5 + \frac{1}{4}s$. This inequality does not cut off (x^*, s^*) . On the other hand, inequality (20) for $x_1 \in [0, 1] \cup \{2\}$ and $x_2 \in [0, 2] \cup [3, 4]$ is $2x_1 + x_2 \leq 6 + \frac{1}{2}s$, which is facet-defining and cuts off (x^*, s^*) . Note that in this case the semi-continuous inequality coincides with the Gomory mixed-integer cut for the basis of (x^*, s^*) .

□

5 Computational Experience

In this section we present our computational results on the effectiveness of semi-continuous cuts for problems with general integer, semi-continuous, and three-term semi-continuous variables. In addition, we compare the effectiveness of branch-and-cut without auxiliary binary variables and the traditional MIP approach for problems with semi-continuous and three-term semi-continuous variables. Our platform consisted of CPLEX 8.0 Callable Library on a Gateway desktop with 1.5 GHz Pentium 4 CPU, 2 Gb RAM, 120 Gb HD, and Windows XP. We kept all CPLEX parameters at their default values, except for preprocessing, which we turned off for all tests on the instances with semi-continuous and three-term semi-continuous variables. The reason that we turned off preprocessing for those tests is that no preprocessing strategy was developed for branch-and-cut without auxiliary binary variables. However, we kept preprocessing on for the tests on the instances with general integer variables. We limited the number of nodes to 1 million in all tests. In any instance there are exclusively integer, semi-continuous, or three-term semi-continuous variables.

The computational results are summarized in Tables 1-4. In the tables we report the number of nodes, computational time, number of CPLEX cuts, number of semi-continuous cuts, and gap improvement. In addition, in Table 1, we report the LP relaxation and the IP optimal values for each instance. The number of CPLEX cuts is the total number of cuts generated automatically by CPLEX throughout the branch-and-bound tree. The CPLEX cuts generated were mainly Gomory cuts, MIR, lifted flow cover, and lifted cover inequalities. On the other

hand, the semi-continuous cuts were generated exclusively at the root node and kept in the cut pool for the remainder of the enumeration process. The gap improvement is given by

$$gapimp = 100 \times \frac{z(lp) - z(root)}{z(lp) - z(ip)},$$

where $z(lp)$, $z(root)$, and $z(ip)$ are the optimal values of the LP relaxation, the LP relaxation with the cuts generated at the root node added, and the IP, respectively.

The instances reported in Tables 2-4 were randomly generated. In Tables 2-4 m denotes the number of knapsack constraints and n denotes the number of integer or semi-continuous variables. We generated 3 instances for each pair $m \times n$. The results reported in Tables 2-4 are the rounded down average results for the instances with the same $m \times n$. The densities of the constraint matrices of these instances range between 15% and 50%. The objective function coefficients are integers uniformly chosen between 10 and 25. The nonzero knapsack coefficients are integers uniformly chosen between 5 and 20. The values of the right-hand-sides are $\lambda \times \max\{\lfloor .3 \sum_{j \in N} a_{ij} \rfloor, \max\{a_{ij} : j \in N\} + 1\}$, where $\lambda = 2$ for the IP instances, and $\lambda = 4$ for the semi-continuous and three-term semi-continuous instances. For the integer variables we chose $u_j = 2$. For the semi-continuous variables we chose $p_j = 0$, $l_j = 3$, and $u_j = 4$. Finally, for the three-term semi-continuous variables we chose $r_j = 1$, $s_j = 2$, $t_j = 3$, and $u_j = 4$.

In Tables 1 and 2 we give the results for the instances with general integer variables. The results in Table 1 refer to 12 instances of the MIPLIB [5]. Because we are interested in problems that contain general integer variables, we changed the domain of the variables in p0033, p0201, p0282, p0548, p2576, and set1ch, which is originally $\{0, 1\}$, to $\{0, 1, 2\}$. These instances are denoted with boldface fonts in Table 1. In Tables 1 and 2, the “sc” columns refer to the tests in which semi-continuous cuts were used (in addition to CPLEX cuts), while the “no sc” columns refer to the tests in which only the CPLEX cuts were used.

In Table 3 we give the results for the instances with semi-continuous variables, and in Table 4 for the instances with three-term semi-continuous variables. In Tables 3 and 4, the “mip” columns refer to the tests in which the traditional MIP approach was used, while the “sc” columns refer to the tests in which branch-and-cut without auxiliary binary variables was used. The “B&B” columns refer to pure branch-and-bound (i.e. no cuts added), while the “B&C” columns refer to branch-and-cut. Finally, the column “cut mip” gives the average number of CPLEX cuts generated for the MIP approach, while the column “cuts sc” gives the average number of semi-continuous cuts generated for branch-and-cut without auxiliary binary variables. In this second case, no integer variables are present in the models, and therefore CPLEX did not generate any cuts.

The branching scheme used for the tests with semi-continuous and three-term semi-continuous variables in branch-and-cut without auxiliary binary variables was the one suggested by Beale [2], and explained in Sect. 1. For the particular case of three-term semi-continuous variables we implemented Beale’s branching

scheme as follows. If $x_j \in (r_j, s_j)$ we add the bounds $x_j \leq r_j$ in one branch and $x_j \geq s_j$ in the other branch. If $x_j \in (s_j, t_j)$ we add the bounds $x_j \leq s_j$ in one branch and $x_j \geq t_j$ in the other branch.

In all cases, semi-continuous cuts were separated as follows. Let x^* be an optimal solution to the LP relaxation that is not feasible for the integer, semi-continuous, or three-term semi-continuous instance that is being solved. We fix any variable x_j for which $x_j^* \in \{0, u_j\}$ at the value of x_j^* , i.e. we take $\tilde{x}_j = x_j^*$. For integer variables, if $x_j^* \notin \mathbb{Z}$, we take $p_j = \lfloor x_j^* \rfloor$ and $l_j = \lceil x_j^* \rceil$. If $x_j^* \in \{0, 1\}$, we take $p_j = 0$ and $l_j = 1$. If $x_j^* = 2$, we take $p_j = 1$ and $l_j = 2$. For three-term semi-continuous variables, if $x_j^* \leq 2$, we take $p_j = 1$ and $l_j = 2$, otherwise we take $p_j = 2$ and $l_j = 3$. To derive the semi-continuous cuts we consider only the knapsack constraints active at x^* . Let $C = \{j \in N : 0 < x_j^* < u_j\}$. If C is a cover, we delete from it as many elements in nondecreasing order of $a_j l_j$ as possible so that the resulting set is still a cover. Note that in the case of IP, C will always be a cover. If the corresponding cover inequality is violated, we lift it approximately, as described in Sect. 3, and we include it in the formulation, otherwise we discard it.

As shown in Tables 1 and 2, semi-continuous cuts can be of considerable use for integer programming problems. Note that CPLEX failed to complete the enumeration within 1 million nodes for rout and set1ch. The computational time and number of cuts reported in the “no sc” columns for these instances are the values when the enumeration was interrupted. In the particular case of (semi-continuous) lifted cover inequalities, the results show that they can be effective for the type of instances tested, i.e. sparse constraint matrices and variables with small upper bounds. In average, semi-continuous cuts reduced the number of nodes in 64% and the computational time in 56% for the MIPLIB instances. For the randomly generated IP instances, the reduction in the number of nodes was 81% and the computational time 68%.

As shown in Tables 3 and 4, branch-and-cut without auxiliary binary variables can be considerably more effective than the traditional MIP approach for problems with semi-continuous, and especially for problems with three-term semi-continuous variables. Note that CPLEX with the MIP approach failed to complete the enumeration within 1 million nodes for the instances with three-term semi-continuous variables for which $m = 70$ and $n = 7,000$. The computational time and number of cuts reported in the “mip” columns for these instances are the values when the enumeration was interrupted. In average, branch-and-cut without auxiliary binary variables reduced the number of nodes in 43% and the computational time in 30% over the MIP approach for the semi-continuous instances. For the three-term semi-continuous instances, the reduction in the number of nodes was 75% and the computational time 80%.

As continued research, we are currently studying semi-continuous cuts other than cover inequalities. In particular, we are evaluating the effectiveness of the simple cuts introduced in [9]. We are also studying the semi-continuous cuts that arise in structures other than the knapsack polyhedron, for example the Simplex

Table 1. Number of nodes, time, and number of cuts for IP: MIPLIB instances

problem	nodes		time		cuts		gapimp		LP	IP
	no sc	sc	no sc	sc	cplex	sc	no sc	sc		
arki001	23,030	11,020	132.21	64	75	127	2	34	7,579,599.81	7,581,040
blend2	1,541	312	1.15	0.05	39	18	19	94	6.92	7.59
flugpl	68	334	0.03	0.02	5	2	47	47	1,167,185.73	1,201,500
gesa2	290	35	1.7	0.02	105	28	84	99	25,476,489.67	25,780,523.02
lseu	586	230	0.11	0.05	21	4	83	69	472.52	700
p0033	314	69	0.04	0.01	13	6	65	93	2,053.71	2,428
p0201	537	131	0.6	0.2	19	4	34	88	6,875	7,615
p0282	2,080	28	1.07	0.12	77	28	81	99	108,732.52	155,146
p0548	277	121	0.4	0.62	121	21	86	95	314.88	1,330
p2576	53,997	317	116.58	3.2	45	12	14	88	2,508	2,592
rout	$10^6 + 507,814$	3,900	2,128		870	38	0.5	90	981.86	1077.56
set1ch	$10^6 + 228,350$	1,533	320		424	38	79	99	32,007.72	54,788.25

Table 2. Number of nodes, time, and number of cuts for IP: random instances

$m \times n$	nodes		time		cuts		gapimp	
	no sc	sc	no sc	sc	cplex	sc	no sc	sc
$20 \times 1,000$	3,024	154	5	2	14	18	54	78
$20 \times 1,500$	3,018	210	8	4	19	29	31	72
$30 \times 2,000$	7,980	312	17	7	19	31	39	82
$30 \times 2,500$	15,230	829	31	12	54	25	41	87
$40 \times 3,000$	32,620	9,730	855	238	98	58	68	72
$40 \times 3,500$	48,750	12,530	2,330	925	134	216	47	89
$50 \times 4,000$	75,520	18,990	4,330	1,022	118	207	52	65
$50 \times 4,500$	112,830	27,620	6,520	1,740	235	198	63	73
$60 \times 5,000$	248,330	31,540	9,278	3,550	334	201	59	87

tableaux and knapsack intersections. We are also testing the effectiveness of semi-continuous cuts for partial-integer and discrete variables.

Very often, semi-continuous constraints arise together with other combinatorial constraints in applications. Perhaps the most frequent one is fixed-charge, which is interesting in its own. We are currently studying a branch-and-cut approach without auxiliary binary variables for problems with fixed-charge constraints.

References

1. E. Balas, “Facets of the Knapsack Polytope,” *Mathematical Programming* 8, 146-164 (1975).
2. E.M.L. Beale, “Integer Programming,” in K. Schittkowski (Ed.), *Computational Mathematical Programming*, NATO ASI F15, Springer-Verlag, 1985, pp. 1-24.

Table 3. Number of nodes, time, and cuts for problems with semi-continuous constraints

$m \times n$	nodes				time				cuts		gapimp	
	mip	sc		mip	sc		mip	sc	mip	sc	mip	sc
		B&B	B&C		B&B	B&C						
30 × 3,000	681	920	680	27	11	22	12	10	45	43		
30 × 3,500	430	780	540	21	7	16	28	90	55	43		
40 × 4,000	2,038	5,327	1,005	31	40	18	79	93	57	88		
40 × 4,500	12,738	22,753	1,729	294	702	39	98	60	53	92		
50 × 5,000	46,873	93,479	41,157	497	1,059	558	170	92	48	49		
50 × 5,500	196,010	403,927	101,092	2,038	3,288	1,570	120	98	65	62		
60 × 6,000	200,746	405,161	184,000	2,199	3,507	2,044	212	99	54	56		
60 × 6,500	180,529	603,289	112,259	3,099	4,798	2,570	249	99	77	71		
70 × 7,000	260,811	344,230	71,358	4,094	3,601	1,758	295	73	59	81		

Table 4. Number of nodes, time, and cuts for problems with three-term semi-continuous constraints

$m \times n$	nodes				time				cuts		gapimp	
	mip	sc		mip	sc		mip	sc	mip	sc	mip	sc
		B&B	B&C		B&B	B&C						
30 × 3,000	421	738	312	31	51	23	28	17	52	39		
30 × 3,500	1,034	641	205	71	52	12	37	54	47	58		
40 × 4,000	12,527	8,310	1,425	640	320	89	96	92	71	81		
40 × 4,500	9,581	4,515	1,130	580	207	71	94	84	68	77		
50 × 5,000	65,491	3,979	1,157	3,108	319	98	217	86	49	86		
50 × 5,500	381,520	203,386	91,010	5,375	1,386	973	415	95	54	70		
60 × 6,000	581,940	109,145	83,678	6,812	1,026	1,052	438	99	48	55		
60 × 6,500	980,529	603,289	259,612	8,408	4,736	2,000	497	88	75	81		
70 × 7,000	$10^6 +$	785,310	317,950	10,804	6,812	2,730	420	97	58	89		

3. E.M.L. Beale and J.A. Tomlin, “Special Facilities in a General Mathematical Programming System for Nonconvex Problems Using Ordered Sets of Variables,” in J. Lawrence (Ed.), *Proceedings of the Fifth International Conference on Operations Research*, Tavistock Publications, 1970, pp. 447-454.
4. D. Bienstock, “Computational Study of a Family of Mixed-Integer Quadratic Programming Problems,” *Mathematical Programming* 74, 121-140 (1996).
5. R.E. Bixby, S. Ceria, C.M. McZeal, and M.W.P. Savelsbergh, “An Updated Mixed Integer Programming Library: MIPLIB 3.0,” *Technical Report TR98-03*, Department of Computational and Applied Mathematics, Rice University (1998).
6. S. Ceria, C. Cordier, H. Marchand, and L.A. Wolsey, “Cutting Planes for Integer Programs with General Integer Variables,” *Mathematical Programming* 81, 201-214 (1998).
7. I.R. de Farias JR., “Semi-continuous Cuts for Mixed Integer Programming,” preprint, Department of Industrial Engineering, University at Buffalo, SUNY (2003).

8. I.R. de Farias JR., E.L. Johnson, and G.L. Nemhauser, "A Generalized Assignment Problem with Special Ordered Sets: A Polyhedral Approach," *Mathematical Programming* 89, 187-203 (2000).
9. I.R. de Farias JR., E.L. Johnson, and G.L. Nemhauser, "Branch-and-Cut for Combinatorial Optimization Problems without Auxiliary Binary Variables," *Knowledge Engineering Review* 16, 25-39 (2001).
10. I.R. de Farias JR., E.L. Johnson, and G.L. Nemhauser "Facets of the Complementarity Knapsack Polytope," *Mathematics of Operations Research* 27, 210-226 (2002).
11. I.R. de Farias JR. and G.L. Nemhauser, "A Polyhedral Study of the Cardinality Constrained Knapsack Problem," *Mathematical Programming* 96, 439-467 (2003).
12. Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh, "Lifted Cover Inequalities for 0-1 Integer Programs: Computation," *INFORMS Journal on Computing* 4, 427-437 (1998).
13. P.L. Hammer, E.L. Johnson, and U.N. Peled, "Facets of Regular 0-1 Polytopes," *Mathematical Programming* 8, 179-206 (1975).
14. T. Ibaraki, T. Hasegawa, K. Teranaka, and J. Iwase, "The Multiple-Choice Knapsack Problem," *Journal of the Operations Research Society of Japan* 21, 59-95 (1978).
15. R.S. Laundy, "Some Logically Constrained Mathematical Programming Problems," *Ph.D. Thesis*, University of Southampton, Southampton, UK (1983).
16. Q. Louveaux and L.A. Wolsey, "Lifting, Superadditivity, Mixed Integer Rounding, and Single Node Flow Sets Revisited," CORE Discussion Paper 2003/1 (2003).
17. G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley-Interscience, 1988.
18. A.F. Perold, "Large-Scale Portfolio Optimization," *Management Science* 30, 1143-1160 (1984).
19. L.A. Wolsey, "Faces for a Linear Inequality in 0-1 Variables," *Mathematical Programming*, 8, 165-178 (1975).

Combinatorial Benders' Cuts

Gianni Codato and Matteo Fischetti

DEI, Department of Information Engineering, University of Padua,
via Gradenigo 6/a, I-35100, Padova, Italy
matteo.fischetti@unipd.it
<http://www.dei.unipd.it/~fisch>

Abstract. Mixed-Integer Programs (MIP's) involving logical implications modelled through big-M coefficients, are notoriously among the hardest to solve. In this paper we propose and analyze computationally an automatic problem reformulation of quite general applicability, aimed at removing the model dependency on the big-M coefficients. Our solution scheme defines a *master* Integer Linear Problem (ILP) with no continuous variables, which contains combinatorial information on the integer-variable feasible combinations that can be “distilled” from the original MIP model. The master solutions are sent to a *slave* Linear Program (LP), which validates them and possibly returns combinatorial inequalities to be added to the current master ILP. The inequalities are associated to minimal (or irreducible) infeasible subsystems of a certain linear system, and can be separated efficiently in case the master solution is integer. This produces an LP relaxation of the master problem which can be considerably tighter than the one associated with original MIP formulation. Computational results on two specific classes of hard-to-solve MIP's indicate the new method produces a reformulation which can be solved some orders of magnitude faster than the original MIP model.

Key words: Mixed-Integer Programs, Benders' Decomposition, Branch and Cut, Computational Analysis.

1 Introduction

We first introduce the basic idea underlying combinatorial Benders' cuts—more elaborated versions will be discussed in the sequel.

Suppose one has a basic 0-1 ILP of the form

$$\min\{c^T x : Fx \leq g, x \in \{0,1\}^n\} \quad (1)$$

amended by a set of “conditional” linear constraints involving additional continuous variables y , of the form

$$x_{j(i)} = 1 \Rightarrow a_i^T y \geq b_i, \text{ for all } i \in I \quad (2)$$

plus a (possibly empty) set of “unconditional” constraints on the continuous variables y , namely

$$Dy \geq e \quad (3)$$

Note that the continuous variables y do not appear in the objective function—they are only introduced to force some feasibility properties of the x 's.

A familiar example of a problem of this type is the classical Asymmetric Travelling Salesman Problem with time windows. Here the binary variables x_{ij} are the usual arc variables, and the continuous variables y_i give the arrival time at city i . Implications (2) are of the form

$$x_{ij} = 1 \Rightarrow y_j \geq y_i + \text{travel_time}(i, j) \quad (4)$$

whereas (3) bound the arrival time at each city i

$$\text{early_arrival_time}(i) \leq y_i \leq \text{late_arrival_time}(i). \quad (5)$$

Another example is the map labelling problem [15], where the binary variables are associated to the relative position of two labels to be placed on a map, the continuous variables give their placement coordinates, and the conditional constraints impose non-overlapping conditions of the type “if label i is placed on the right of label j , then the placement coordinates of i and j must obey a certain linear inequality giving a suitable separation condition”.

The usual way implications (2) are modelled within the MIP framework is to use the (in)famous *big-M method*, where large positive coefficients M_i are introduced to activate/deactivate the conditional constraints as in:

$$a_i^T y \geq b_i - M_i(1 - x_{j(i)}) \text{ for all } i \in I \quad (6)$$

This yields a (often large) mixed-integer model involving both x and y variables—whereas, in principle, y variables are just artificial variables. Even more importantly, due to the presence of the big-M coefficients, the LP relaxation of the MIP model is typically very poor. As a matter of fact, the x solutions of the LP relaxation are only marginally affected by the addition of the y variables and of the associated constraints. In a sense, the MIP solver is “carrying on its shoulders” the burden of *all* additional constraints and variables in (2)-(3) at *all* branch-decision nodes, while they becomes relevant only when the corresponding $x_{j(i)}$ attains value 1 (typically, because of branching).

Of course, we can get rid of the y variables by using Benders' decomposition [3], but the resulting cuts are weak and still depend on the big-M values. As a matter of fact, the classical Benders' approach can be viewed as a tool to speed-up the solution of the LP relaxation, but not to improve its quality.

The idea behind “combinatorial” Benders' cuts is to work on the space of the x -variables only, as in the classical Benders's approach. However, we model the additional constraints (2)-(3) through the following *Combinatorial Benders'* (CB) cuts:

$$\sum_{i \in C} x_{j(i)} \leq |C| - 1 \quad (7)$$

where $C \subseteq I$ induces a *Minimal (or Irreducible) Infeasible Subsystem* (MIS, or IIS, for short) of (2)-(3), i.e., any inclusion-minimal set of row-indices of system (2) such that the linear subsystem

$$SLAVE(C) := \begin{cases} a_i^T y \geq b_i, \text{ for all } i \in C \\ Dy \geq e \end{cases}$$

has no feasible (continuous) solution y .

A CB cut is violated by a given $x^* \in [0, 1]^n$ if and only if $\sum_{i \in C} (1 - x_{j(i)}^*) < 1$. Hence the corresponding separation problem essentially consists of the following steps: (i) weigh each conditional constraint $a_i^T y \leq b_i$ in (2) by $1 - x_{j(i)}^*$; (ii) weigh each unconditional constraint in (3) by 0; and (iii) look for a minimum-weight MIS of the resulting weighted system—a NP-hard problem [1,10].

A simple polynomial-time heuristic for CB-cut separation is as follows. Given the (fractional or integer) point x^* to be separated, start with $C := \{i \in I : x_{j(i)}^* = 1\}$, verify the infeasibility of the corresponding linear subsystem $SLAVE(C)$ by classical LP tools, and then make C inclusion-minimal in a greedy way. Though extremely simple, this efficient separation turns out to be exact when x^* is integer.

The discussion above suggests the following exact Branch & Cut solution scheme. We work explicitly with the integer variables x only. At each branching node, the LP relaxation of a *master problem* (namely, problem (1) amended by the CB cuts generated so far) is solved, and the heuristic CB separation is called so as to generate new violated CB cuts (and to assert the feasibility of x^* , if integer).

The new approach automatically produces a sequence of CB cuts, which try to express in a purely-combinatorial way the feasibility requirement in the x space—the CB cut generator acting as an automatic device to distill more and more combinatorial information from the input model. As a consequence of the interaction among the generated CB cuts, other classes of combinatorial cuts are likely to be violated, hence allowing other cut separators to obtain a further improvement. We found that the $\{0, \frac{1}{2}\}$ -cuts addressed in [5,2] fit particularly well in this framework, and contribute substantially to the overall efficacy of the approach.

In the present paper we aim at investigating whether the above method can be useful to approach certain types of MIP's which are notoriously very hard to solve. As shown in the computational section, this is indeed the case: even in its simpler implementation, on some classes of instances the new approach allows for a speed-up of some orders of magnitude with respect to ILOG-Cplex 8.1, one of the best MIP solvers on the market.

It is worth noting that, using the new technique, the role of the big-M terms in the MIP model vanishes—only implications (2) are relevant, no matter the way they are modelled. Actually, the approach suggests an extension of the MIP

modelling language where logical implications of the type (2) can be stated explicitly in the model. Moreover, the correctness of the method only relies on the capability of detecting, at run time, minimal infeasible configurations of the binary variables (the so-called *nogoods* in Constraint Programming), to be translated in terms of CB cuts. In this respect, our technique goes in the direction of integrating Constraint Programming and Mathematical Programming, in the spirit of the recent papers of Hooker and Ottosson [12] and Thorsteinsson [18]. With respect to the previous proposals, however, our approach has the important feature of basing the search for nogoods on the feasibility of certain linear systems, thus providing an elegant and quite general framework for its efficient implementation. Moreover, as shown in Section 3, our approach has interesting connections with Chvátal's *resolution search* [7] and with Glover-Tangedhal's *dynamic branch and bound* [11].

The paper is organized as follows. In Section 2 we present the new approach in a more general context, whereas in Section 3 we discuss its connections with previous methods from the literature. As already stated, our CB cut separator requires a fast determination of MIS's: this important topic is addressed in Section 4, where an approach particularly suited to our application is described. Computational results are presented in Section 5, with the aim of verifying whether a simple implementation of the new method can already produce improved performance with respect to the application of a sophisticated MIP solver such as ILOG-Cplex 8.1 (at least, on some problem classes which fit particularly well in our scheme). Finally, some conclusions are drawn in Section 6.

The present paper is based on the master thesis of the first author [8], which was awarded the 2003 Camerini-Carraresi prize by the Italian Operation Research association (AIRO).

2 Combinatorial Benders' Cuts

Let P be a MIP problem with the following structure:

$$P : \quad z^* := \min \quad c^T x + d^T y \quad (8)$$

$$\text{s.t.} \quad Fx \leq g \quad (9)$$

$$Mx + Ay \geq b \quad (10)$$

$$Dy \geq e \quad (11)$$

$$x_j \in \{0, 1\} \quad \text{for } j \in B \quad (12)$$

$$x_j \text{ integer} \quad \text{for } j \in G \quad (13)$$

where x is a vector of integer variables, y is a vector of continuous variables, G and B are the (possibly empty) index sets of the general-integer and binary variables, respectively, and M is a matrix with exactly one nonzero element for every row i , namely the one indexed by column $j(i) \in B$. In other words, we assume the linking between the integer variables x and the continuous variables y is only due to a set of constraint of the type

$$m_{i,j(i)}x_{j(i)} + a_i^T y \geq b_i \text{ for all } i \in I \quad (14)$$

where all variables $x_{j(i)}$, $i \in I$, are binary.

We consider the case $d = 0$ first, i.e., we assume the MIP objective function does not depend on the continuous variables, and leave case $d \neq 0$ for a later analysis. In this situation, we can split problem P into two sub-problems:

– *MASTER*:

$$z^* = \min \quad c^T x \quad (15)$$

$$\text{s.t.} \quad Fx \leq g \quad (16)$$

$$x_j \in \{0, 1\} \text{ for } j \in B \quad (17)$$

$$x_j \text{ integer for } j \in G \quad (18)$$

– *SLAVE*(\tilde{x}), a linear system parametrized by \tilde{x} :

$$Ay \geq b - M\tilde{x} \quad (19)$$

$$Dy \geq e \quad (20)$$

Let us solve the master problem at integrality. If this problem turns out to be infeasible, the P also is. Otherwise, let x^* be an optimal solution (we exclude the unbounded case here, under the mild assumption that, e.g., the general-integer variables are bounded).

If the linear system *SLAVE*(x^*) has a solution, say y^* , then clearly (x^*, y^*) is an optimal solution of P . If the slave is infeasible, instead, x^* itself is infeasible for problem P . We therefore look for a MIS of *SLAVE*(x^*), involving the rows of A indexed by C (say), and observe that at least one binary variable $x_{j(i)}$ has to be changed in order to break the infeasibility. This condition can be translated by the following linear inequality in the x space, that we call Combinatorial Benders' (CB) cut:

$$\sum_{i \in C: x_{j(i)}^* = 0} x_j + \sum_{i \in C: x_{j(i)}^* = 1} (1 - x_j) \geq 1. \quad (21)$$

One or more CB cuts of this type are generated in correspondence of a given infeasible x^* , and added to the master problem. Iterating the procedure produces an exact solution method in the spirit of Benders' decomposition.

Of course, it is advisable to exploit the CB cuts within a modern Branch & Cut scheme which hopefully produces violated cuts at each node of the branching tree—and not just in correspondence of an integer optimal solution of the master. Note that the correctness of the Branch & Cut method only requires the generation of a violated CB cut (if any) before each updating of the incumbent solution of the master, i.e., any heuristic CB separator that guarantees to be exact for an *integer* x^* already suffices to get a valid solution method.

We now address the case $c = 0$ and $d \neq 0$, arising when the objective function only depends on the continuous variables y . In this situation, we cannot accommodate the objective function into the master problem. Instead, we can add the

bound inequality $d^T y \leq UB - \epsilon$ to the slave system, where UB is the value of the incumbent solution, and ϵ is a sufficiently small positive value. In this way, the CB cuts will translate both the feasibility and the optimality requirements. More specifically, at the iterations where $SLAVE(x^*)$ (amended by the bound inequality) is infeasible, we can generate one or more violated CB cuts, as required. At the iterations where this system is feasible, instead, we can find an optimal solution y^* of the LP problem $\min\{d^T y : Ay \geq b - Mx^*, Dy \geq e\}$ and update the best incumbent solution by (x^*, y^*) . The overall method will stop when the current master (that now looks for a feasible x improving the incumbent) becomes infeasible.

Finally, we observe that the case $c \neq 0$ and $d \neq 0$ cannot be dealt with effectively by our method. A possible approach is to make an external binary search of the (unknown) value of $d^T y^*$ in an optimal solution (x^*, y^*) of P , and exploit this information by introducing bound constraints of the type $d^T y = d^T y^*$ into the slave. However, this naive approach would imply the solution of a (possibly long) series of MIP's, hence its practical effectiveness should be investigated computationally—this topic is left to future research.

3 Problem Reformulation and Connections with Chvátal's Resolution Search

At first glance, the required assumptions—in particular, (14)—restrict the range of applicability of our method considerably. However, there are several important (and difficult) MIP's which fit naturally in our scheme. Some of them will be addressed in the computational section.

In addition, there is a simple reformulation method that can extend its applicability significantly. The idea is to introduce a (continuous) copy x_j^c of each binary variable x_j , $j \in B$, and to link the two copies through the constraints:

$$x_j = x_j^c \quad \text{for } j \in B \tag{22}$$

By construction, the above constraints can play the role of (14), thus linking the master problem to the slave.

In particular, one can always reformulate a generic MIP with no general-integer variables (i.e., involving only binary and continuous variables) as follows. Introduce the continuous copy x^c of the entire vector x . The initial master takes the binary variable vector x , plus the constraints $Fx \leq g$ along with the obvious constraints $x \in \{0, 1\}^n$; the master objective function is zero (or $c^T x$ if $d = 0$). The slave keeps the continuous variable vectors x^c and y , the constraints $Mx^c + Ay \geq b$, $Dy \geq e$ along with the linking equations $x = x^c$, plus the objective function bound $c^T x^c + d^T y \leq UB - \epsilon$, where UB is the value of the incumbent solution.

Actually, one can even decide to remove $Fx \leq g$ from the master, and to introduce $Fx^c \leq g$ into the slave (the master objective function being zero). With this choice, at each iteration the master only contains the distilled combinatorial

information (notably, CB cuts) that exclude certain configurations of the binary variables—because they are infeasible or cannot lead to an improved solution. The master then iteratively detects new binary solutions x^* which fulfill the master constraints generated so far, and invokes the slave for validation. The slave verifies the feasibility of the proposed x^* (with respect to the LP relaxation of the original MIP, amended by the upper bound constraint), possibly updates the incumbent, and then returns one or more CB cuts related to some forbidden *minimal* configurations of the binary variables.

This process resembles closely Chvátal's resolution search [7]. Roughly speaking, resolution search can be viewed as an attempt to get rid of the rigid tree paradigm used within enumeration schemes. Convergence of generic enumerative methods for combinatorial optimization problems requires to record information about the subsets of the solution space that have been already explored—so as to avoid considering twice a same subset, and to abort the search when no unexplored subset exists. For each subset, an *oracle* (borrowing Chvátal terminology) is invoked to answer the basic question: Is there any hope that this subset contains a solution improving the incumbent? If (possibly after having updated the incumbent) the answer is “no”, the subset is discarded. If the answer is “yes”, the set is subdivided further. For MIP problems, the subsets are typically defined by a *branching strategy*, i.e., by fixing the value of certain subsets of the integer variables.

The familiar *implicit enumeration* scheme is then a possible implementation of the above search strategy, obtained when the explored solution subsets form a nested family, each subset being represented by a node in the branching tree. The branching tree is now viewed as a ordered (but rigid) way to subdivide the solution space and to feed the oracle. For each subset, the LP relaxation of the MIP model is used to implement the oracle. In case of a “no” answer, the current subset is typically discarded with no further analysis—in particular, no attempt is made to enlarge it.

Resolution search can be viewed as a different way to control the solution process. Here, the branching tree is replaced by a pool of logical conditions that summarize the previous computation. At each iterations, a partial assignment of the binary variables is found, which fulfills the logical conditions in the current pool. If no such assignment exists, then the enumeration is over and the incumbent is a provable optimal solution. Otherwise, the oracle is invoked in two distinct greedy phases. In the *waxing* phase, the current partial assignment is iteratively extended up to a point where the oracle returns a “no” answer (with the incumbent possibly updated). In the subsequent *waning* phase, the current partial assignment is iteratively reduced as long as the oracle answer remains “no”. At the end of the waning phase, the current partial assignment corresponds to an *obstacle*, i.e., to a minimal set of variables that have to be changed in order to get a “yes” answer from the oracle. The logical condition “change the value at least one of the variables in the obstacle” is then added to the pool of logical conditions, so as to avoid to face the same obstacle in the later exploration, and the process is iterated. The analogy with our approach

is now evident: our master problem plays the role of a 0-1 ILP model for the logical conditions in the pool, whereas the slave is our implementation of the oracle function to detect one or more obstacles efficiently.

Even without the reformulation above, BC cuts can be generated within a standard Branch & Cut solution framework not based explicitly on the master/slave decomposition we propose. For the sake of notation, let us consider a pure 0-1 ILP of the form $\min\{c^T x : Ax \geq b, x \in \{0,1\}^n\}$. Take a generic branching node $NODE_h$ which is fathomed by the classical lower bound criterion, possibly after the updating of the incumbent solution. The node corresponds (say) to the fixing $x_j = x_j^*$ for some $j \in J_h$, where $x_j^* \in \{0,1\}$ is the known branching value for variable x_j at $NODE_h$. The fathoming condition then implies that the slave linear system made by the “linking equations” $x_j = x_j^*$ for $j \in J_h$ and by the inequalities $Ax \geq b, x \in [0,1]^n, c^T x \leq UB - \epsilon$, is infeasible. Finding a MIS of this system then yields an inclusion-minimal $C \subseteq J_h$ whose associate BC cut (21) can be added to the current formulation. (Of course, in this context the new cut is only useful if C is a proper subset of J_h , so as to exclude a solution subset strictly larger than the one associated with the current node $NODE_h$.) This approach tends to limit a known drawback of standard Branch & Cut codes, namely “depending heavily on the branching strategy used to select the next variable to branch on and its value” [7]. It can be viewed as a simple yet (hopefully) effective method for “branching resequencing” in the spirit of the *Dynamic Branch and Bound* of Glover and Tangedhal [11].

4 Fast MIS Search

In this section we describe an efficient algorithm to find a MIS of an infeasible linear system, that fits particularly well within our solution approach. The method is in the spirit of the one discussed by Parker and Ryan in [16]

MIS search can be formulated as follows: given an infeasible system of inequalities, say $\tilde{A}y \geq \tilde{b}$, find an inclusion-minimal set of its rows yielding an infeasible system.

We therefore construct the LP

$$\min \quad 0^T y \tag{23}$$

$$\text{s.t. } \tilde{A}y \geq \tilde{b} \tag{24}$$

and its dual

$$\max \quad u^T \tilde{b} \tag{25}$$

$$\text{s.t. } u^T \tilde{A} = 0^T \tag{26}$$

$$u \geq 0 \tag{27}$$

It is known that if the primal problem is infeasible, then the corresponding dual can be either unbounded or infeasible. Now, dual infeasibility is excluded by the fact that $u = 0$ is always dual feasible, hence primal infeasibility corresponds

to dual unboundedness, i.e., to the existence of a dual solution u^* such that $u^{*T}\tilde{b} > 0$ (hence ku^* for a sufficiently large $k > 0$ is a feasible dual solution with arbitrarily large objective value). Therefore we can replace the dual objective function by the following constraint:

$$u^{T\tilde{b}} = 1 \quad (28)$$

This modified dual problem is meant at finding a linear combination of the rows of $\tilde{A}y \geq \tilde{b}$ leading to a valid inequality $u^{*T}\tilde{A}y \geq u^{*T}\tilde{b}$ with all-zero left-hand side coefficients and strictly positive right-hand side, thus proving the infeasibility of $\tilde{A}y \geq \tilde{b}$ —the existence of u^* is guaranteed, for infeasible systems, by the Farkas' lemma.

It is known [10] that each *vertex* of the dual polyhedron defined by (26)–(28) has a support defining an MIS (whereas minimality is not guaranteed for an arbitrary point u of the same polyhedron). In our applications, we look for a solution u^* having a small support in the set of the linking constraints (19), whereas we do not actually care on the minimality with respect to (20). This suggests the use of the heuristic dual objective function $\sum_i w_i u_i$, where weights w_i 's are used to drive the LP solver to select a solution u^* with the desired characteristics. Moreover, by iteratively setting to zero some of the u variables we can easily detect alternative MIS's, which is very important in order to generate several violated BC cuts at each call of the separation heuristic.

5 Computational Results

To evaluate its effectiveness, we implemented our method in C++ and embedded it within the ILOG-Cplex Concert Technology 1.2 framework, based on ILOG-Cplex 8.1 [13,14].

In our implementation, we read an input MIP in the form (8)–(13), and verify that (a) the linking constraints are of the form (14) with binary $x_{j(i)}$, and (b) the objective function only depends on the integer variables. If the MIP does not fulfill the above conditions, we simply invoke ILOG-Cplex on the original MIP—in this preliminary version, no attempt is made to reformulate the MIP. Otherwise, we construct automatically the master/slave decomposition, and invoke the ILOG-Cplex solver on the master. During the ILOG-Cplex Branch & Cut execution, we call our separation procedures for CB and $\{0, \frac{1}{2}\}$ -cuts. In the CB separation, the integer components of the current master LP solutions x^* are sent to the slave, in the attempt of generating one or more CB cuts through the MIS search described in Section 3. As to $\{0, \frac{1}{2}\}$ -cuts, we use the separation code of Andreello, Caprara, and Fischetti [2]. This separation proved quite effective, and allowed for a reduction of up to 50% of the computing time of our method for some instances of the test-bed. (Observe that effective $\{0, \frac{1}{2}\}$ -cuts cannot be derived from the original MIP formulation of the instances in our test-bed, due to the presence of the continuous variables.)

In order not to overload the LP formulation, we avoided calling our CB and $\{0, \frac{1}{2}\}$ -cut separators at each node of the branching tree, but applied them (a)

before each updating of the incumbent solution; (b) at each node with a tree depth not greater than 10; and (c) after each backtracking step (see [2] for a discussion on the use of this type of strategies).

Due to the heuristic nature of our separation procedures for CB and $\{0, \frac{1}{2}\}$ -cuts, and since the number of generated cuts tends to increase steeply, all cuts are stored in a constraint pool, which is purged from time to time.

It should be observed that ILOG-Cplex 8.1 does not implement an internal cut pool: once a (globally valid) cut has been generated at run time, it is added statically to the LP formulation and never removed. An important exception (that we exploit in our code) arises for locally-valid cuts, which are automatically removed from the formulation the first time they are no longer guaranteed to be valid (because of a backtracking step); removed cuts are not saved.

The lack of a native pool within ILOG-Cplex 8.1 is not really an issue in itself, in that one can easily implement an external data structure to store the CB and $\{0, \frac{1}{2}\}$ -cuts. A possible issue is however the impossibility of removing a (globally valid) cut from the current LP. This makes it impossible to purge the current LP by removing some of its constraints, with the risk it becomes soon too large. A possible work-around is to define the CB and $\{0, \frac{1}{2}\}$ -cuts as *local* (as opposed to global) cuts, so they are automatically removed from time to time from the LP. Though this approach does not allow us to have a complete control on the cut removal mechanism (nor on the size of the LP), it works reasonably well in practice—although it implies a certain memory overhead.

5.1 The Test-Bed

The set of problems we have used can be divided into two categories.

Map Labelling. As mentioned in the introduction, this problem consists in placing as many rectangular labels as possible (without overlap) in a given map. If placed, a label has a limited degree of movement around its associated “reference point” (a pre-specified point of the map corresponding to, e.g., the city whose name is in the label). This problem has been formulated as a MIP model involving big-M coefficients by Müzsel and Klau [15], who report very poor results when trying to solve the model directly. Much better results are in fact obtained by the same authors when using a different (purely combinatorial) 0-1 ILP model, where the binary variables are associated with arcs of a suitable digraph, and the non-overlapping conditions are translated by rank inequalities forbidding the choice of all the arcs in a circuit with certain properties. Actually, it was precisely the nice correspondence between the MIP model and its graph reformulation that motivated us to generalize the Müzsel-Klau construction, thus originating the work presented in the present paper. As a matter of fact, applying our method to their MIP model produces automatically a slave problem with a network structure, hence it can be associated with a digraph whose circuits (under appropriate conditions) correspond to minimal infeasible subsystems—and hence to CB cuts.

Our test-bed contains 18 map labelling instances, of the so-called 4-slider (4S) and 4-position (4P) type, kindly provided by G.W. Klau.

As shown in the computational section, the results we obtained on map labelling problems are comparable with those reported in [15], even though our method is more general and does not exploit the specific network structure of the slave. This is an indication that the overhead introduced in our implementation (in particular, for computing MIS's via LP methods rather than using fast graph-search algorithms) is acceptable.

Two-Group Statistical Classification (discriminant analysis). This problem can be described briefly as follow; see, e.g., Rubin [17] for more details. We have a population whose members can be divided into two distinct classes—for example, people affected or not by a certain disease. We can measure a number of characteristics that are related to the partition above—e.g., biological parameters that show the presence or absence of the disease. Based on the available measures, we want to obtain a linear function which allows us to decide whether a new member belongs to the first or second class. More specifically, we are looking for a linear function that minimize the number of misclassified members in the known sample¹. This problem can be modelled as a MIP problem with big-M coefficients: the unknowns are the coefficients of the linear function, and for every member there is a binary variable used to deactivate the inequality in case of misclassification. The purpose is to minimize the number of deactivated inequalities, so the objective function is just the sum of the binary variables. (In [17] a slightly different MIP model is used, involving also a real-valued variables in the objective; this model can easily be transformed into the one we use in the present study.)

The raw data from which we have generated our instances has been taken from a public archive maintained at UCI [19], and converted to the final MIP model with a slightly modified version of a program kindly provided us by P. A. Rubin. This resulted into 38 instances coming from different real situations (i.e., disease classification, interpretation of physical phenomena, animal behavior, etc.).

5.2 Results

Our experiments have been performed on a PC AMD Athlon 2100+ with 1 GByte RAM, and GNU/Linux (kernel 2.4) Operating System.

All the instances of our test-bed have been processed twice: the first time by solving the original MIP model through the commercial ILOG-Cplex 8.1 solver (with default settings), and the second by using our implementation of

¹ Designing optimal linear classifiers consists in minimizing (resp., maximizing) the number of misclassified (resp., correctly classified) members. Given the infeasible inequality system imposing correct classification of all members, the problem amounts to deleting the minimum number of inequalities to make the system feasible, or equivalently to finding a maximum feasible subsystem (MaxFS); see, e.g., [6,1].

the master/slave decomposition based on CB cuts and $\{0, \frac{1}{2}\}$ -cuts (still based on the ILOG-Cplex 8.1 library). A time-limit of 3 CPU hours was imposed for each run. In addition, we set the `IloCplex::NodeFileInd` parameter to 3: this forces Cplex to store the active-node data into the hard disk when physical memory is exhausted (due to the efficiency of this mechanism, we had no memory problems even when handling branching trees much larger than 1GB)

In the tables below, label *CBC* refers to our code (based on CB cuts), whereas label *Cplex* refers to the standard ILOG-Cplex 8.1 solver. We compared the two codes in terms of execution times and/or integrality gaps computed with respect to the best integer solution found, on the following three instance subclasses:

Subset 1	File name	Cplex		CBC		Statistics	
		opt.	Time hh : mm : ss	Time hh : mm : ss	t.Cplex/ t.CBC	t.CBC/ t.Cplex	
STAT. ANALYSIS							
Chorales-116	24	1 : 24 : 52	0 : 10 : 18	8.2	12.1%		
Balloons76	10	0 : 00 : 10	0 : 00 : 14	71.4	1.4%		
BCW-367	8	0 : 08 : 33	0 : 00 : 13	39.4	2.5%		
BCW-683	10	2 : 02 : 29	0 : 00 : 32	229.7	0.4%		
WPBC-194	5	0 : 57 : 17	0 : 03 : 32	16.2	6.2%		
Breast-Cancer-400	6	0 : 02 : 50	0 : 00 : 16	1062	0.1%		
Glass-163	13	0 : 56 : 17	0 : 00 : 05	675.4	0.1%		
Horse-colic-151	5	0 : 04 : 50	0 : 00 : 23	12.6	7.9%		
Iris-150	18	0 : 09 : 29	0 : 01 : 10	8.1	12.3%		
Credit-300	8	0 : 19 : 35	0 : 00 : 02	587.5	0.2%		
Lymphography-142	5	0 : 00 : 11	0 : 00 : 01	11.0	9.1%		
Mech-analysis-107	7	0 : 00 : 05	0 : 00 : 01	5.0	20.0%		
Mech-analysis-137	18	0 : 07 : 44	0 : 00 : 27	17.2	5.8%		
Monks-tr-122	13	0 : 02 : 05	0 : 00 : 05	25.0	4.0%		
Pb-gr-txt-198	11	0 : 04 : 21	0 : 00 : 05	52.2	1.9%		
Pb-pict-txt-444	7	0 : 02 : 07	0 : 00 : 02	63.5	1.6%		
Pb-hl-pict-277	10	0 : 04 : 17	0 : 00 : 27	9.5	10.5%		
Postoperative-88	16	0 : 15 : 16	0 : 00 : 01	916.0	0.1%		
BV-OS-282	6	0 : 05 : 13	0 : 00 : 24	13.0	7.7%		
Opel-Saab-80	6	0 : 01 : 03	0 : 00 : 13	4.8	20.6%		
Bus-Van-437	6	0 : 09 : 17	0 : 00 : 28	19.9	5.0%		
HouseVotes84-435	6	0 : 04 : 59	0 : 00 : 11	27.2	3.7%		
Water-treat-206	4	0 : 01 : 10	0 : 00 : 06	11.7	8.6%		
Water-treat-213	5	0 : 17 : 00	0 : 00 : 51	20.0	5.0%		
MAP LABELLING							
CMS 600-1	600	1 : 08 : 41	0 : 04 : 34	15.0	6.6%		
TOTALS		8 : 29 : 51	0 : 24 : 11	21.1	5%		

Table 1. Problems solved to proven optimality by both Cplex and CBC

1. instances solved to proven optimality by both CBC and Cplex
2. instances solved to proven optimality by CBC but not by Cplex
3. instances not solved (to proven optimality) by CBC nor by Cplex

Notice that there was no instance in our test-bed that was solved by Cplex but not by CBC.

For the first subset, Table 1 reports the instance names, the optimal objective values (*opt.*), the Cplex and CBC computing times, and their ratios. On these instances, CBC turns out to be up to 3 orders of magnitude faster than Cplex. According to the final row of the table, Cplex ran for almost 8.5 hours to solve all the instances of this subset, while CBC requires about 24 minutes, i.e., the latter code was 21 times faster in processing the whole subset.

In Table 2 we report the number of branching nodes enumerated by Cplex and by CBC for the exact solution of the instances of the first subset. Note

File name (Subset 1)	n. nodes Cplex	n. nodes CBC
STAT. ANALYSIS		
Chorales-116	10,329,312	20,382
Balloons-76	40,481	4
BCW-367	79,980	463
BCW-683	399,304	671
WPBC-194	806,188	26,439
Breast-cancer-400	181,990	1
Glass-163	3,412,702	64
Horse-colic-151	135,018	2,184
Iris-150	970,659	1,290
Credit-300	176,956	66
Lymphography-142	8,157	106
Mech-analysis-107	11,101	68
Mech-analysis-137	938,088	1,888
Monks-tr-122	262,431	357
Pb-gr-txt-198	135,980	110
Pb-pict-txt-277	71,031	1,026
Pb-hl-pict-444	22,047	115
Postoperative-88	2,282,109	171
BV-OS-282	56,652	1,044
Opel-Saab-80	87,542	7,314
Bus-Van-437	55,224	6,795
HouseVotes84-435	42,928	734
Water-treat-206	12,860	482
Water-treat-213	168,656	4,036
MAP LABELLING		
CMS 600-1	110,138	14

Table 2. Number of branch-decision nodes enumerated by Cplex and by CBC, respectively

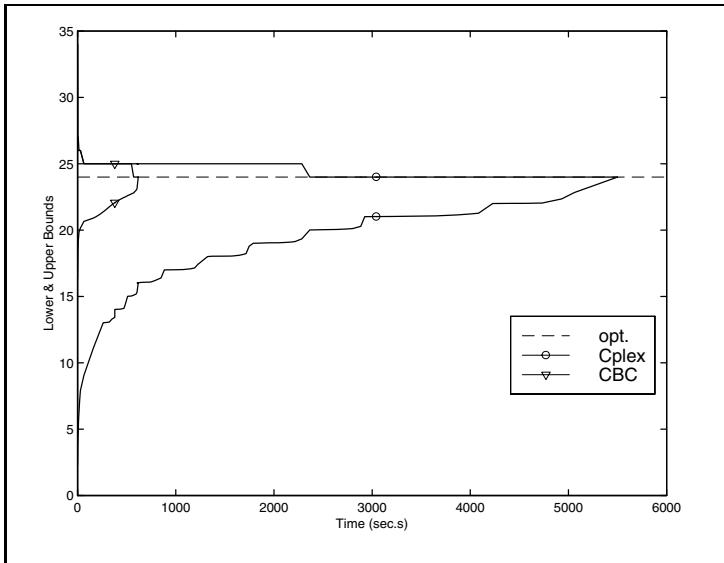


Fig. 1. Optimizing the statistical-analysis instance Chorales-116 (minimization problem)

however that a direct comparison of these figures is not immediate in that the two methods actually work on different formulations—hence processing a single node may require different computing times.

Figure 1 reports the graph of the best bound and of the incumbent solution values for Cplex and CBC executions on a sample instance in this subset.

The results for the second subset of instances are given in Table 3 (statistical analysis) and in Table 4 (map labelling). As in Table 1, for CBC we report the optimization time and the optimal objective value, while for Cplex we give the value of the incumbent solution (*best sol.*) and of the best bound (*best bound*), their percentage gap (*Gap%*, computed as $100 |best_sol - best_bound|/best_sol$), as well as the MBytes of memory used (*mem. MB*). For each instance, we report two distinct rows for Cplex execution: the first one refers to the time instant when CBC finished its computation, and the second row gives the situation when Cplex is aborted (time limit expired). Finally, in the last two rows of the table we report the total computing time and the average Cplex gaps, computed both when CBC finished its execution (*same t.*), and at the end of its own computation (*end*). Figure 2 gives an illustration of the Cplex and CBC behavior on a sample instance of this second subset. For this class, the difference between CBC and Cplex is even more evident: not only Cplex was not able to solve any of these problems within the imposed time limit, but the best-bound trend suggests it is very unlikely it will converge even allowing for an extremely large computing time. On the whole, CBC solved all the statistical analysis instances in less than 2 hours while, after the same time, Cplex had an average gap of more than 70%.

Subset 2		Cplex					CBC	
File Name		Time hh:mm:ss	best sol.	best bound	Gap %	mem MB	Time hh:mm:ss	opt.
Chorales-134	0: 36 :23	33		16.0	51%	371	0: 36 :23	30
	3: 00 :58	30		21.1	30%	992		
Chorales-107	0: 04 :19	28		12.1	57%	61	0: 04 :19	27
	3: 01 :27	27		22.2	18%	711		
Breast-Cancer-600	0: 00 :13	108		1.5	99%	9	0: 00 :13	16
	3: 00 :11	16		13.2	18%	45		
Bridges-132	0: 03 :39	33		5.1	85%	44	0: 03 :39	23
	3: 01 :09	23		10.0	56%	1406		
Mech-analysis-152	0: 34 :12	22		12.1	45%	328	0: 34 :12	21
	3: 00 :50	21		16.1	24%	865		
Monks-tr-124	0: 01 :55	27		8.1	70%	25	0: 01 :55	24
	3: 00 :35	24		20.0	17%	381		
Monks-tr-115	0: 04 :16	29		9.1	69%	67	0: 04 :16	27
	3: 01 :07	27		19.0	30%	1131		
Solar-flare-323	0: 00 :04	51		5.0	90%	18	0: 00 :04	38
	3: 00 :45	43		17.0	61%	977		
BV-OS-376	0: 09 :04	9		3.1	65%	9	0: 09 :04	9
	3: 00 :10	9		6.0	33%	56		
BusVan-445	0: 10 :31	13		3.0	77%	11	0: 10 :31	8
	3: 00 :06	9		5.1	43%	56		
TOTALS			Gap (same t.)	71%			1: 44 :36	
		30: 07 :18	Gap (end)	33%				

Table 3. Statistical Analysis problems solved to proven optimality by CBC but not by Cplex

Moreover, we found that after 28 additional hours of computation the gap would have been only halved. An analogous behavior was observed on map labelling instances, for which the gap is not even halved after more than 31 additional hours of computation.

The results for the third subset are summarized in the full paper [9]: in all cases, CBC obtained much smaller gaps than Cplex at the end of the allowed computing time. An illustrative graph is given in Figure 3.

6 Conclusions

We have proposed and analyzed computationally an automatic MIP reformulation method, aimed at removing the model dependency on the big-M coefficients often used to model logical implications. Computational results on two specific classes of hard-to-solve MIP's (namely, *Statistical Analysis* and *Map Labelling* problems) show that the new method produces automatically a reformulation which can be solved some orders of magnitude faster than the original MIP model.

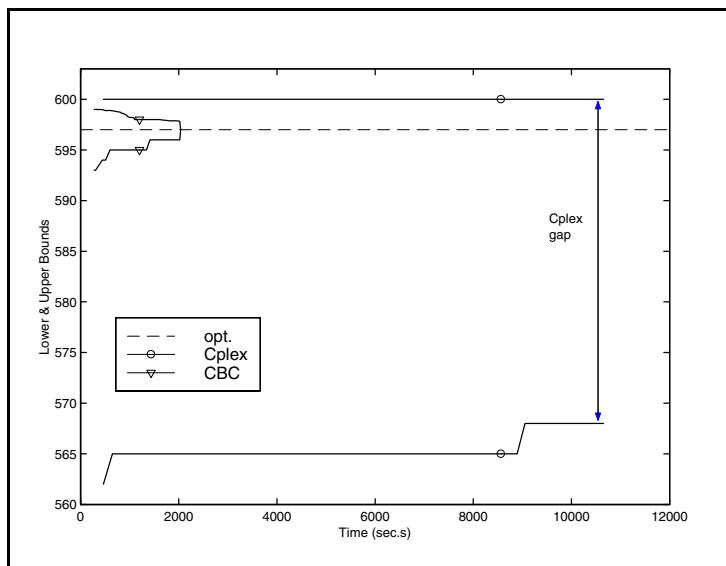


Fig. 2. Optimizing the map labelling instance CMS600-1 (4P) (maximization problem)

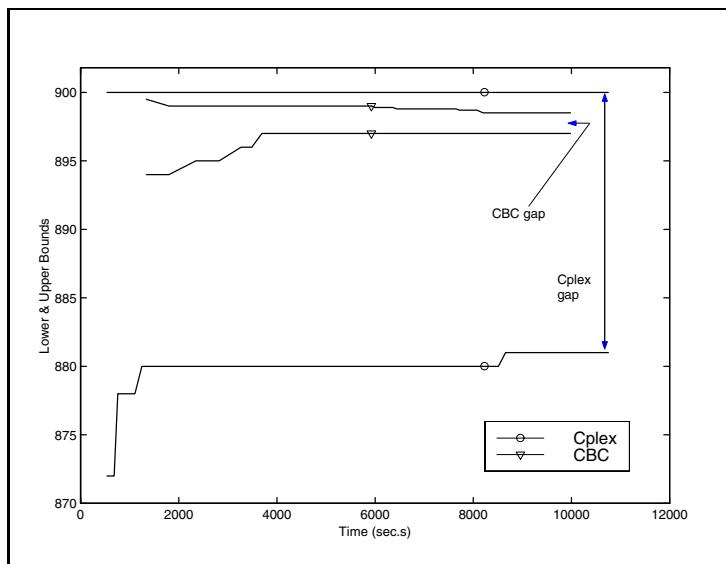


Fig. 3. Optimizing the map labelling instance CMS900-0 (4S) (maximization problem)

Subset 2 File Name	Cplex					CBC	
	Time hh:mm:ss	best sol.	best bound	Gap %	mem. MB	Time hh:mm:ss	opt.
CMS 600-0 (4S)	0: 04 :27	592	600	1.35%	18	0: 04 :27	600
	3: 03 :00	594	600	1.01%	770		
CMS 650-0 (4S)	0: 06 :26	638	650	1.88%	20	0: 06 :26	649
	3: 02 :34	646	650	0.62%	480		
CMS 650-1 (4S)	0: 04 :50	647	650	0.46%	7	0: 04 :50	649
	3: 03 :13	648	650	0.31%	904		
CMS 700-1 (4S)	0: 13 :06	686	700	2.04%	58	0: 13 :06	699
	3: 03 :00	691	700	1.30%	1045		
CMS 750-1 (4S)	0: 07 :53	738	750	1.63%	28	0: 07 :53	750
	3: 02 :19	741	750	1.21%	521		
CMS 750-4 (4S)	0: 07 :05	736	750	1.90%	28	0: 07 :05	748
	3: 00 :24	743	750	0.94%	417		
CMS 800-0 (4S)	0: 19 :15	773	800	3.49%	55	0: 19 :15	798
	3: 02 :16	773	800	3.49%	533		
CMS 800-1 (4S)	0: 22 :24	784	800	2.04%	92	0: 22 :24	800
	3: 02 :30	786	800	1.78%	761		
Railway	0: 00 :31	95	103	8.42%	1	0: 00 :31	100
	3: 00 :02	100	101	1.00%	19		
CMS 600-0 (4P)	0: 00 :01	543	600	10.5%	2	0: 00 :04	600
	3: 02 :57	574	600	4.53%	782		
CMS 600-1 (4P)	0: 39 :07	565	600	6.19%	184	0: 39 :07	597
	3: 02 :55	568	600	5.63%	831		
TOTALS	33: 25 :10	Gap (same t.)	3.6%			2: 05 :4.1	
		Gap (end)	2.0%				

Table 4. Map Labelling problems solved to proven optimality by CBC but not by Cplex

Future direction of work should address the more general case where the MIP objective function depends on both the continuous and the integer variables, and analyze computationally the merits of the resulting technique. Some preliminary results in this directions (based on the theory reported in Section 3) are encouraging.

Acknowledgements

Work marginally supported by MIUR and CNR, Italy.

References

1. E. Amaldi, M.E. Pfetsch and L.E. Trotter, “On the Maximum Feasible Subsystem Problem, IISs and IIS-Hypergraphs”, Mathematical Programming 95, 3, 533–554, 2003.

2. G. Andreello, A. Caprara, M. Fischetti, "Embedding Cuts within a Branch & Cut Framework: a Computational Study with $\{0, \frac{1}{2}\}$ -cuts", Technical Report, DEI, University of Padova, 2003.
3. J. F. Benders, "Partitioning Procedures for Solving Mixed Variables Programming Problems", Numerische Mathematik 4, 238–252, 1962.
4. R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, R. Wunderling, "MIP: Theory and Practice—Closing the Gap", available at www.ilog.com/products/optimization/tech/research/mip.pdf.
5. A. Caprara, M. Fischetti, " $\{0, \frac{1}{2}\}$ - Chvátal-Gomory Cuts", Mathematical Programming 74, 221–235, 1996.
6. J. Chinneck, "Fast Heuristics for the Maximum Feasible Subsystem Problem", INFORMS J. Comput. 13 (3), 210–223, 2001.
7. V. Chvátal, "Resolution Search", Discrete Applied Mathematics 73, 81–99, 1997.
8. G. Codato, "A Combinatorial Approach to Benders' Decomposition", Tesi di Laurea, University of Padova, 2003 (in italian).
9. G. Codato, M. Fischetti "Combinatorial Benders' Cuts", full paper available at www.dei.unip.it/~fisch/cbc.pdf
10. J. Gleeson and J. Ryan, "Identifying Minimally Infeasible Subsystems of Inequalities", ORSA Journal on Computing 2 (1), 61–63, 1990.
11. F. Glover and L. Tangedhal, "Dynamic Strategies for Branch and Bound", Omega 4 (5), 571–576, 1976.
12. J.N. Hooker, G. Ottosson, "Logic-based Benders Decomposition", Mathematical Programming, Mathematical Programming, 96, 33–60, 2003.
13. ILOG Cplex 8.1: User's Manual and Reference Manual, ILOG, S.A., <http://www.ilog.com/>, 2003.
14. ILOG Concert Technology 1.2: User's Manual and Reference Manual, ILOG, S.A., <http://www.ilog.com/>, 2003.
15. G. W. Klau, P. Mütsel, "Optimal Labelling of Point Features in Rectangular Labelling Models", Mathematical Programming Ser. B, vol. 94 (2-3), 435–458, 2003.
16. M. Parker, J. Ryan, "Finding the Minimum Weight IIS Cover of an Infeasible System of Linear Inequalities", Ann. Math. Artificial Intelligence 17, 107–126, 1996.
17. P.A. Rubin, "Solving Mixed Integer Classification Problem by Decomposition", Annals of Operations Research 74, 51–64, 1997.
18. E.S. Thorsteinsson, "Branch-and-Check: A Hybrid Framework Integrating Mixed Integer Programming and Constraint Logic Programming", Seventh International Conference on Principles and Practice of Constraint Programming (CP2001), 2001.
19. P.M. Murphy, D. W. Aha, "UCI Repository of Machine Learning Databases", University of California, Department of Information and Computer Science, Irvine, CA, 1994; available at <http://www.ics.uci.edu/~mlearn/MLRepository.html>

A Faster Exact Separation Algorithm for Blossom Inequalities

Adam N. Letchford¹, Gerhard Reinelt², and Dirk Oliver Theis²

¹ Department of Management Science, Lancaster University,
Lancaster LA1 4YW, England

A.N.Letchford@lancaster.ac.uk

² Institut für Informatik, University of Heidelberg, Germany
Gerhard.Reinelt@informatik.uni-heidelberg.de

Dirk.Theis@informatik.uni-heidelberg.de

Abstract. In 1982, Padberg and Rao gave a polynomial-time separation algorithm for b -matching polyhedra. The current best known implementations of their separation algorithm run in $\mathcal{O}(|V|^2|E| \log(|V|^2/|E|))$ time when there are no edge capacities, but in $\mathcal{O}(|V||E|^2 \log(|V|^2/|E|))$ time when capacities are present.

We propose a new exact separation algorithm for the capacitated case which has the same running time as for the uncapacitated case. For the sake of brevity, however, we will restrict our introduction to the case of perfect 1-capacitated b -matchings, which includes, for example, the separation problem for perfect 2-matchings. As well as being faster than the Padberg-Rao approach, our new algorithm is simpler and easier to implement.

Key Words: matching, polyhedra, separation.

1 Introduction

Let $G = (V, E)$ be an undirected graph, and let $b \in \mathbb{Z}_+^{|V|}$ be a vector of vertex capacities with $\sum_{v \in V} b_v$ even. A (*perfect 1-capacitated*) b -matching is a set of edges, such that for each $i \in V$, there are exactly b_i edges in the set incident to i . If we define for each edge e the integer variable x_e , which is one if e appears in the matching and zero otherwise, then the incidence vectors of b -matchings are the solutions of:

$$x(\delta(i)) = b_i, \text{ for all } i \in V \quad (1)$$

$$0 \leq x_e \leq 1, \text{ for all } e \in E \quad (2)$$

$$x_e \in \{0, 1\}, \text{ for all } e \in E. \quad (3)$$

Here, as usual, $\delta(i)$ represents the set of vertices incident to i , and $x(F)$ denotes $\sum_{e \in F} x_e$.

The convex hull in $\mathbb{R}^{|E|}$ of solutions of (1)–(3) is called the *b*-matching polytope. Edmonds and Pulleyblank (see Edmonds [2] and Pulleyblank [13]) gave a

complete linear description of this polytope. It is described by the *degree equalities* (1), the *bounds* (2) and the following so-called *blossom inequalities*:

$$x(\delta(W) \setminus F) - x(F) \geq 1 - |F|, \quad \text{for all } W \subset V, F \subseteq \delta(W) \text{ with } b(W) + |F| \text{ odd.} \quad (4)$$

Here $\delta(W)$ represents the set of edges with exactly one end-vertex in W and $b(W)$ denotes $\sum_{i \in W} b_i$. The set W is sometimes called the *handle* and the edges in the set F are called *teeth* of the blossom inequalities.

In their seminal paper, Padberg and Rao [10] devised a combinatorial polynomial-time *separation algorithm* for b -matching polytopes. A separation algorithm is a procedure which, given a vector $x^* \in \mathbb{R}^{|E|}$ lying outside of the polytope, finds a linear inequality which is valid for the polytope yet violated by x^* (see for example [6]). Separation algorithms are at the heart of the well known *branch-and-cut* approach (see [11]) to combinatorial optimization.

Clearly, testing if a degree equation or bound is violated can be performed in $\mathcal{O}(|E|)$ time, so the main contribution of [10] is to identify violated blossom inequalities.

The Padberg-Rao separation algorithm involves up to $|V| + |E| - 1$ maximum flow computations on a special graph, the so-called *split graph*, which has $|V| + |E|$ vertices and $2|E|$ edges. Using the pre-flow push algorithm [3], this leads to a worst-case running time of $\mathcal{O}(|E|^3 \log |V|)$, or $\mathcal{O}(|V|^6 \log |V|)$ in dense graphs, for solving the separation problem for blossom inequalities.

Grötschel and Holland [5] observed that the above-mentioned max-flow computations can in fact be carried out on graphs with only $\mathcal{O}(|V|)$ vertices and $\mathcal{O}(|E|)$ edges. Although the idea behind this is simple, it reduces the overall running time to $\mathcal{O}(|V||E|^2 \log(|V|^2/|E|))$, which is $\mathcal{O}(|V|^5)$ in the case of a dense graph.

In this paper we give a separation algorithm for blossom inequalities which has running time $\mathcal{O}(|V|^2|E|\log(|V|^2/|E|))$, or $\mathcal{O}(|V|^4)$ in dense graphs. As well as being faster than the Padberg-Rao and Grötschel-Holland approaches, it is simpler and easier to implement.

Besides for matching problems, the algorithm is also important for solving the *Traveling Salesman Problem* (TSP). The special blossom inequalities obtained when $b_i = 2$, for all i , are valid for the TSP, and facet-inducing under mild conditions (see Grötschel and Padberg [7,8]). Thus we obtain a faster separation algorithm for the TSP as a by-product. In fact, the algorithm is applicable to a general class of cutting planes for integer programs, called $\{0, \frac{1}{2}\}$ -Chvátal-Gomory cuts, see Caprara and Fischetti [1].

2 Review of the Original Algorithm

For the purposes of clarity, we will briefly review the original algorithm.

Let x^* be a given vector to be separated and suppose that it satisfies the degree equalities and bounds. Padberg and Rao proceed as follows. Take the

graph G and divide each edge e into two halves by adding an extra vertex, which we call *splitting vertex*, and denote by k_e . One half (which we shall call the *normal half*) is given weight x_e^* , whereas the other half (which we shall call the *flipped half*) is given weight $1 - x_e^*$. The resulting graph $\widehat{G} = (\widehat{V}, \widehat{E})$ is called the *split graph*. It has $|\widehat{V}| = |V| + |E|$ vertices and $|\widehat{E}| = 2|E|$ edges.

Next, Padberg and Rao assign parities to the vertices. For $i \in V$, let f_i denote the number of flipped edges which are incident to the vertex i of \widehat{G} . A vertex is labelled *even* or *odd* according to whether $b_i + f_i$ is an even or odd number. It is not difficult to see that there is a violated blossom inequality (4) if and only if there exists an odd cut of weight strictly less than 1 in the split graph. Here a cut $\delta(U')$ is called odd if each of its two shores U' and $\overline{U'}$ contains an odd number of odd vertices. If $\delta(U')$ is an odd cut in the split graph with $\hat{x}(\delta(U')) < 1$, then $U' \cap V$ is the handle of a violated blossom inequality.

This construction enabled Padberg and Rao [10] to use their generic minimum weight odd cut algorithm. This algorithm involves running the classical *cut-tree* algorithm of Gomory and Hu [4] on the split graph, with the odd vertices as terminals. This means that a maximum flow algorithm must be run several times on the split graph (or on certain graphs obtained from the split graph by contraction of edges). As mentioned in the introduction, the number of odd vertices is at least $|E|$ and at most $|V| + |E|$ leading to an overall running time of $\mathcal{O}(|E|^3 \log |V|)$, or $\mathcal{O}(|V|^6 \log |V|)$ in the case of a dense graph.

Grötschel and Holland's Improvement

As mentioned in the introduction, a significant improvement to this algorithm was made by Grötschel and Holland [5]. (In fact, they were only interested in 2-matchings, but the same argument applies in the general case.) Consider an arbitrary edge $e = \{i, j\} \in E$. It is obvious that the maximum amount of flow which can be directly sent from i to j in the split graph is equal to the minimum of x_e^* and $1 - x_e^*$. From this observation it follows that prior to computing a max flow, for every edge $e \in E$, one of the two edges incident to the vertex $k_e \in \widehat{V}$ can be contracted. This means that the max-flow problems can be conducted on graphs which are similar in size to G (only the source and sink vertices can be in $\widehat{V} \setminus V$). However, the number of max-flows which have to be computed is still at least $|E|$ and at most $|V| + |E|$, leading to an overall running time of $\mathcal{O}(|V||E|^2 \log(|V|^2/|E|))$, which is $\mathcal{O}(|V|^5)$ in a dense graph.

3 The New Algorithm

In this section we give an algorithm which separates blossom inequalities in time $\mathcal{O}(|V|^2|E| \log(|V|^2/|E|))$. In particular, we will need only $|V| - 1$ max-flow computations, and these can be performed in the original graph G .

The basic idea of the algorithm is to compute a cut-tree for G with terminal vertex set V and weights w , where w is defined by letting, for every edge $e \in E$

$$w_e := \min(x_e^*, 1 - x_e^*).$$

Algorithm 1 Blossom_Separation

Input: Graph G and vector x^* .
Output: A violated blossom inequality, if one exists.

- 1: Compute a cut-tree for G with weights $w := \min(x_e^*, 1 - x_e^*)$ and V as set of terminal vertices by using any cut-tree algorithm.
- 2: **For** each of the $|V| - 1$ cuts $\delta(W)$ stored in the cut-tree **do**
- 3: Find the best set F of teeth for the candidate handle W , i.e., among all sets $F \subseteq \delta(W)$ with $b(W) + |F|$ odd, find one which minimizes the value $x^*(\delta(W) \setminus F) + |F| - x^*(F)$.
- 4: **If** $x^*(\delta(W) \setminus F) + |F| - x^*(F) < 1$ **then**
- 5: Output the violated blossom inequality. **Stop**.
- 6: **End if**
- 7: **End for**
- 8: Output “There exists no violated blossom inequality.”

Then, we take each of the $|V| - 1$ cuts $\delta(W)$ stored in the cut-tree and consider W as a candidate for the handle of a violated blossom inequality. For each candidate handle we find the best set of teeth, and thus decide if there exists a violated blossom inequality with this handle. Our blossom separation algorithm is displayed as algorithm 1. (Note that the algorithm can be modified to find the most violated blossom inequality.)

Finding the best set of teeth (step 3) can be done by sorting (see [5]), which would take $\mathcal{O}(|E| \log |E|)$ time (which is sufficiently fast to achieve the improved running time of the algorithm). It is possible to reduce the time for step 3 to $\mathcal{O}(|E|)$ by the following easy observation (this fact is implicit in [12]). Let F denote the set of edges $e \in \delta(W)$ with $1 - x_e^* < x_e^*$. If $b(W) + |F|$ is odd, then the minimum is achieved by this set F . Otherwise, let $f \in \delta(W)$ be the edge which minimizes the term $\max(x_e^*, 1 - x_e^*) - \min(x_e^*, 1 - x_e^*)$ over all $e \in \delta(W)$. If we define F' to be the symmetric difference of the sets F and $\{f\}$, then $b(W) + |F'|$ is odd, and F' is an optimal set of teeth for the handle W .

Step 3 has to be carried out at most $|V|$ times, leading to a running time of $\mathcal{O}(|V||E|)$ for the loop in steps 2–7. Thus, the running time is dominated by the computation of the $|V| - 1$ max-flows in step 1.

In Section 5, we will prove that algorithm 1 solves the separation problem for blossom inequalities (4). Before we do that, for the purpose of clarity, we repeat some facts about cut-trees and minimum odd cuts in the next section.

4 Cut-Trees and Minimum Odd Cuts

The contents and terminology of this section refer to graphs, cut-trees and minimum odd cuts in general. If $G = (V, E)$ is a graph with positive edge weights and $p, q \in V$ are distinct vertices, then we denote by $\lambda(p, q) := \lambda_G(p, q)$ the weight of a minimum (p, q) -cut in the graph G with respect to the edge weights.

The Gomory-Hu Algorithm

Given a graph with positive edge weights and a set of *terminal vertices* $T \subseteq V$, a *cut-tree* is a tree whose nodes, called *supernodes*, are sets which partition the vertex set of G . Each supernode contains precisely one terminal vertex which is called its *representative*. The edges of the cut-tree are weighted. If A and B are two adjacent supernodes and r_A and r_B are their respective representatives, then the weight of the edge $\{A, B\}$ of the cut-tree equals $\lambda(r_A, r_B)$, the value of the maximum (r_A, r_B) -flow in G .

Since the supernodes form a partition of the vertex set of G , every edge of the cut-tree defines a cut in G , because its removal defines a partition of V into two sets. We say that the cut-tree edge *induces* this cut. In the definition of a cut-tree, we require that, if A and B are two adjacent supernodes and r_A and r_B are their respective representatives, then the edge $\{A, B\}$ of the cut-tree induces a minimum (r_A, r_B) -cut in G .

Gomory and Hu [4] gave an algorithm which produces a cut-tree by performing $|T| - 1$ max-flow computations. Their algorithm starts with a (trivial) cut-tree with terminal vertex set $\{r\}$ and successively performs the following tree expansion step, which enlarges the terminal vertex set.

Tree Expansion. Suppose that \mathcal{C} is a cut-tree with terminal vertex set $T \subsetneq V$, and let $t \in V \setminus T$ be contained in the supernode R of \mathcal{C} , whose representative is $r \in T$. Let S_1, \dots, S_l be the neighbors of R in \mathcal{C} , and for $i = 1, \dots, l$ let $\delta(U_i)$ be the cut induced by the edge $\{R, S_i\}$ of \mathcal{C} , where $S_i \subseteq U_i$. Construct the graph G_R by identifying in G , for each $i = 1, \dots, l$, the set U_i to a single vertex s_i , where loops are deleted and parallel edges are merged while adding their weights. Now let $\delta(X)$ be a minimum (r, t) -cut in the graph G_R . Construct a new tree \mathcal{C}' out of \mathcal{C} by replacing the supernode R of \mathcal{C} with two supernodes $R \cap X$ and $R \setminus X$, which are linked by an edge with weight $\lambda_{G_R}(r, t)$. Then for each edge $\{R, S_i\}$ of \mathcal{C} add to \mathcal{C}' an edge $\{R \cap X, S_i\}$, if $s_i \in X$, or an edge $\{R \setminus X, S_i\}$, if $s_i \notin X$, while keeping the original weight.

Theorem 1 ([4]). \mathcal{C}' is a cut-tree for G with terminal vertex set $T \cup \{t\}$.

Knowledge of this expansion step will be relevant in Section 5. For completeness, we also note the uncrossing lemma of Gomory and Hu [4].

Lemma 1 ([4]). *With the above terminology, there exists a minimum (r, t) -cut $\delta(Y)$ in G , which does not cross any of the cuts $\delta(U_i)$, $i = 1, \dots, l$, i.e., which satisfies $U_i \subseteq Y$ or $U_i \subseteq \overline{Y}$ for all $i = 1, \dots, l$. In particular $\lambda_{G_R}(r, t) = \lambda_G(r, t)$ holds.*

The Padberg-Rao Min-Odd-Cut Algorithm

Now we assume that $|T|$ is an even number. The elements of T are called *odd* vertices. A cut $\delta(U)$ in G is called *odd*, if $|T \cap U|$ is an odd number. The following theorem of Padberg and Rao [10] relates minimum odd cuts and cut-trees and gives rise to their minimum odd cut algorithm.

Theorem 2 ([10]). Let \mathcal{C} be a cut-tree for G with terminal vertex set T . We call a cut in \mathcal{C} odd, if both shores contain an odd number of supernodes.

Any minimum odd cut in \mathcal{C} induces a minimum odd cut in G .

Clearly, a minimum odd cut in a tree must consist of a single edge. Thus there exists at least one edge of the cut-tree which induces a minimum odd cut.

In the next section, we will need the following lemma.

Lemma 2. Suppose that the value of a minimum odd cut in G is strictly less than 1. Let \mathcal{C}' be a cut-tree with a certain terminal vertex set $T' \subseteq V$. Denoting by \mathcal{S}' the set of supernodes of \mathcal{C}' , we define the following set of odd supernodes:

$$\{S \in \mathcal{S}' \mid |T \cap S| \text{ odd}\}.$$

Now, we call a cut in \mathcal{C}' odd, if each of its shores contains an odd number of odd supernodes.

If $\lambda(p, q) \geq 1$ for every two vertices $p \in T$ and $q \in T'$ which are both contained in the same supernode of \mathcal{C}' , then any minimum odd cut in \mathcal{C}' induces a minimum odd cut in G .

Proof. First we note that the argument of Padberg and Rao [10] which proves theorem 2 still works, if the terminal vertex set of the cut-tree is bigger than the set of odd vertices. This means that if \mathcal{C}'' is a cut-tree for G with terminal vertex set $T'' \supset T$, and we call a supernode of \mathcal{C}'' odd if its representative is odd, then any minimum odd cut of \mathcal{C}'' induces a minimum odd cut in G .

Thus we consider a cut-tree \mathcal{C}'' which is obtained from \mathcal{C}' by performing tree expansion until the set of terminal vertices is enlarged from T' to $T' \cup T$. Because of what we have just noted, one of the edges of \mathcal{C}'' induces a minimum odd cut in G . A minimum odd cut of G cannot separate two vertices $p \in T$ and $q \in T'$ which are contained in the same supernode of \mathcal{C}' , because $\lambda(p, q) \geq 1$ and the value of a minimum odd cut in G is strictly less than 1. Neither can it separate two vertices $p_1, p_2 \in T$, which are contained in the same supernode of \mathcal{C}' , because, if q is the representative of this supernode,

$$\lambda(p_1, p_2) \geq \min(\lambda(p_1, q), \lambda(q, p_2)) \geq 1.$$

So none of the edges which have been created by the tree expansion from \mathcal{C}' to \mathcal{C}'' induces a minimum odd cut in G . Hence, there must exist an edge in \mathcal{C}' , which induces one.

By the easy fact that by the definition of the set of odd supernodes of \mathcal{C}' , an edge of \mathcal{C}' induces an odd cut in G if and only if it is an odd cut in \mathcal{C}' , the claim of the lemma follows.

5 Proof of Correctness of the New Algorithm

Let T denote the set of vertices of the split graph \hat{G} which are odd according to the Padberg-Rao labelling.

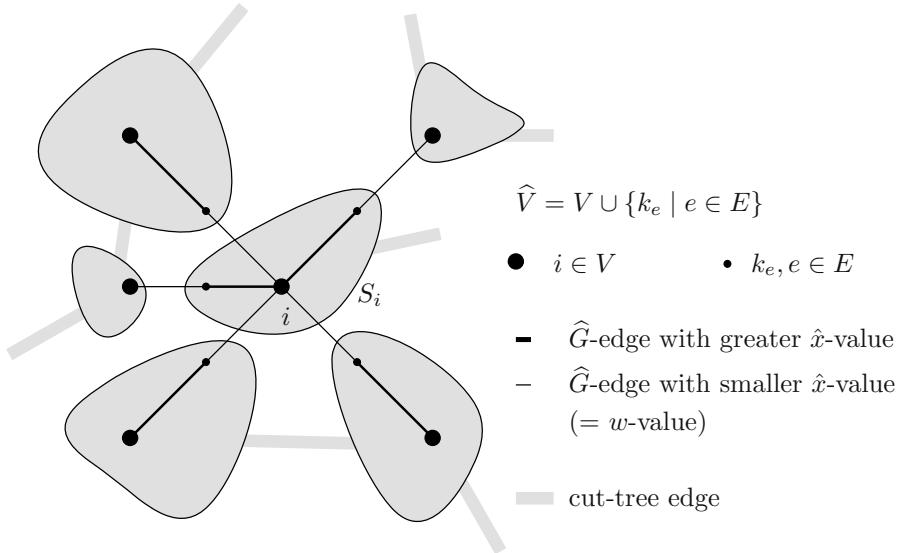


Fig. 1. A partition of \hat{V} based on \mathcal{C}_G resulting in a cut-tree for \hat{G} with set of terminal vertices $V \subset \hat{V}$.

To prove the correctness of algorithm 1, we will first make clear that we can think of any cut-tree \mathcal{C}_G of G with terminal vertex set V as a cut-tree of \hat{G} with terminal vertex set $V \subset \hat{V}$. Then we will show how a cut-tree of \hat{G} containing a minimum odd cut might look like, if the value of the minimum odd cut is strictly less than 1. From this we will deduce that among the cuts $\delta(W)$ stored in the cut-tree \mathcal{C}_G for G , there is one which is the handle of a violated blossom inequality, if one exists.

We need some notation. For every edge $e = \{i, j\} \in E$ define

$$\phi(e) := \begin{cases} i, & \text{if } \hat{x}_{\{i, k_e\}} \geq \hat{x}_{\{k_e, j\}}, \\ j, & \text{if } \hat{x}_{\{j, k_e\}} > \hat{x}_{\{k_e, i\}}, \end{cases}$$

i.e., $\phi(e)$ is the end vertex of e to which k_e is linked by the \hat{G} -edge with *bigger* \hat{x} -weight.

Let \mathcal{C}_G denote a cut-tree of G with weights w and set of terminal vertices V . Define a partition $\widehat{\mathcal{S}}$ of the set $\hat{V} = V \cup \{k_e | e \in E\}$ by letting, for each $i \in V$,

$$S_i := \{i\} \cup \left\{ k_e \in \hat{V} \mid e \in \delta_G(i) \wedge \phi(e) = i \right\},$$

which means we include a vertex k_e of \hat{G} corresponding to an edge e of G into that set S_i represented by the neighbor i of k_e to which it is linked by the \hat{G} -edge with *greater* \hat{x} -value, i.e., $\hat{x}_{k_e, i} \geq \hat{x}_{k_e, j} = w_e$. See Fig. 1.

Apparently, these sets define a partition of the vertex set \widehat{V} of \widehat{G} . Let $\mathcal{C}_{\widehat{G}}$ denote the tree which results from replacing for every $i \in V$ the supernode $\{i\}$ of \mathcal{C}_G by the supernode S_i , while keeping the edges and edge weights. Then we have the following fact (without proof).

Lemma 3. *$\mathcal{C}_{\widehat{G}}$ is a cut-tree of \widehat{G} with weights \widehat{x} and set of terminal vertices V .*

If we wanted to solve the blossom separation problem by computing a minimum odd cut in \widehat{G} , we could start with \mathcal{C}_G , replace the supernodes as just described, and then follow the Gomory-Hu procedure to expand the cut-tree until the terminal vertex set is $V \cup T$. Because of Lemma 2 (with $T' := V \cup T$), we could find a minimum odd cut in this cut-tree.

We will see now that this is not necessary. Instead, we will just show how a cut-tree which contains a minimum odd cut in \widehat{G} might look like. To do this, we simulate tree expansion steps to a point when the condition of Lemma 2 applies.

First note that $T \setminus V$ only contains split vertices. From Lemma 2, we know that, if $k_e \in T$ is contained in the supernode together with $v \in V \cup T$, we need to consider the minimum (k_e, v) -cut only if its value $\lambda_{\widehat{G}}(k_e, v)$ is strictly less than 1. As the proof of the following proposition shows, in this case, a minimum (k_e, v) -cut can be obtained by modifying a cut from the cut-tree, and the update of the cut-tree is easy to predict.

Proposition 1. *Define the following subset of the (odd) vertices of \widehat{G} :*

$$Q := \{k_e \mid e \in E, \lambda_{\widehat{G}}(k_e, \phi(e)) < 1\}.$$

There is a Gomory-Hu cut-tree \mathcal{C}' of \widehat{G} with terminal vertex set $V \cup Q$ which is a subdivision of $\mathcal{C}_{\widehat{G}}$.

To be more precise, every edge $\{S_i, S_j\}$ is replaced by a path

$$S'_i, \{k_{e_1}\}, \dots, \{k_{e_r}\}, S'_j, \quad r \geq 0$$

of supernodes, where the $\{k_{e_h}\}$ are supernodes of \mathcal{C}' consisting of a single vertex $k_{e_h} \in Q$, and we let $S'_i := S_i \setminus Q$ and $S'_j := S_j \setminus Q$.

Proof. The proof is by induction. We will perform expansion steps on a sequence of cut-trees, which have the property that for each $e \in E$ the vertex k_e is either contained in the supernode whose representative is $\phi(e)$, or it is itself a singleton supernode $\{k_e\}$, subdividing (with some other singleton supernodes maybe) an edge of $\mathcal{C}_{\widehat{G}}$.

The condition clearly holds for the first cut-tree in the sequence, i.e., before performing any tree expansion steps, namely $\mathcal{C}_{\widehat{G}}$.

For the induction step we perform tree expansion. For the notation, see the paragraph about tree expansion in Section 4 above. We choose an arbitrary $k_e \in Q$ which is not a representative. By induction hypothesis, this implies that it is contained in a supernode R whose representative is $\phi(e) \in V$. We need to know how a minimum $(\phi(e), k_e)$ -cut in G_R looks. Suppose that $e = \{\phi(e), j\}$,

i.e., $j \in V$ is the other end of the edge e of G . One of the sets U_1, \dots, U_l , say U_i , contains the vertex j .

Claim. We claim that

$$\delta(\{k_e, s_i\})$$

is a minimum $(\phi(e), k_e)$ -cut in G_R .

If the claim is correct, then the tree expansion step produces the new supernodes $R \setminus \{k_e\}$ and $\{k_e\}$, thus making $\{k_e\}$ a singleton supernode. The other split vertices remain untouched, so the induction step is completed.

Proof of the claim. By the uncrossing lemma of Gomory and Hu (Lemma 1), we know that $\lambda_G(\phi(e), k_e) = \lambda_{G_R}(\phi(e), k_e)$. Thus, because $\lambda(\phi(e), k_e) < 1$ and $\hat{x}_{\{\phi(e), k_e\}} + \hat{x}_{\{k_e, j\}} = 1$, we know that j and $\phi(e)$ are on different shores of any minimum $(\phi(e), k_e)$ -cut in G_R . This means that any minimum $(\phi(e), k_e)$ -cut in G_R is also a $(\phi(e), s_i)$ -cut, and this of course remains true if we move the vertex k_e to the $\phi(e)$ -shore of such a cut. From this it follows that

$$\lambda_{G_R}(\phi(e), s_i) \leq \lambda_{G_R}(\phi(e), k_e) - \hat{x}_{\{\phi(e), k_e\}} + \hat{x}_{\{k_e, j\}}.$$

It is clear from the definition of a cut-tree that $\delta(s_i)$ is a minimum $(\phi(e), s_i)$ -cut in G_R . So we obtain

$$\begin{aligned} \lambda_{G_R}(\phi(e), k_e) &\geq \lambda_{G_R}(\phi(e), s_i) + \hat{x}_{\{\phi(e), k_e\}} - \hat{x}_{\{k_e, j\}} \\ &= \delta(s_i) + \hat{x}_{\{\phi(e), k_e\}} - \hat{x}_{\{k_e, j\}} \\ &= \delta(\{k_e, s_i\}). \end{aligned}$$

Thus the claim is true and the proof of the proposition is finished.

We are now ready to complete the proof of correctness of algorithm 1.

Suppose there exists a violated blossom inequality. This implies that the value of the minimum odd cut in \widehat{G} is strictly less than 1. From Lemma 2, we know that a minimum odd cut of \widehat{G} is stored in any cut-tree with terminal vertex set $V \cup Q$ —in particular in the cut-tree \mathcal{C}' whose existence is proven in the proposition. Let $\delta(U')$ be a minimum odd cut of \widehat{G} induced by an edge of \mathcal{C}' .

If we let $W := U' \cap V$, then there exists a set $F \subseteq \delta(W)$ such that $x(\delta(W) \setminus F) - x(F) \geq 1 - |F|$ is a blossom inequality which is violated by x^* . But from the proposition we know that the cut $\delta(W)$ is induced by an edge of \mathcal{C}_G , the cut-tree which is used in algorithm 1. Hence the set W (or \overline{W}) is one of those which are considered as candidate handles in Step 3 of algorithm 1, so the algorithm finds this (or another) violated inequality.

Hence the new algorithm is correct.

Remark 1. Our algorithm can be easily adapted to more general u -capacitated b -matching problems, perfect or otherwise. Details will be given in the full version of this paper [9].

References

1. Caprara, A., Fischetti, M.: $\{0, \frac{1}{2}\}$ -Chvátal-Gomory cuts. *Math. Program.* **74** (1996) 221–235.
2. Edmonds, J.: Maximum matching and a polyhedron with 0-1 vertices. *J. Res. Nat. Bur. Standards* **69B** (1965) 125–130.
3. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum flow problem. *J. of the A.C.M.* **35** (1988) 921–940.
4. Gomory, R.E., Hu, T.C.: Multi-terminal network flows. *SIAM J. Applied Math.* **9** (1961) 551–570.
5. Grötschel, M., Holland, O.: A cutting plane algorithm for minimum perfect 2-matching. *Computing* **39** (1987) 327–344.
6. Grötschel, M., Lovász, L., Schrijver, A.J.: *Geometric Algorithms and Combinatorial Optimization*. Wiley, New York (1988).
7. Grötschel, M., Padberg, M.W.: On the symmetric travelling salesman problem I: inequalities. *Math. Program.* **16** (1979) 265–280.
8. Grötschel, M., Padberg, M.W.: On the symmetric travelling salesman problem II: lifting theorems and facets. *Math. Program.* **16** (1979) 281–302.
9. Letchford, A.N., Reinelt, G., Theis, D.O.: Odd minimum cut-sets and b -matchings revisited. (in preparation)
10. Padberg, M.W., Rao, M.R.: Odd minimum cut-sets and b -matchings. *Math. Oper. Res.* **7** (1982) 67–80.
11. Padberg, M.W., Rinaldi, G.: Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Oper. Res. Lett.* **6** (1987) 1–7.
12. Padberg, M.W., Rinaldi, G.: Facet identification for the symmetric traveling salesman polytope. *Math. Program.* **47** (1990) 219–257.
13. Pulleyblank, W.R.: Faces of matching polyhedra. Ph.D thesis, University of Waterloo. (1973)

LP-based Approximation Algorithms for Capacitated Facility Location

Retsef Levi^{1*}, David B. Shmoys^{2**}, and Chaitanya Swamy^{3**}

¹ School of ORIE, Cornell University, Ithaca, NY 14853.
`r1227@cornell.edu`

² School of ORIE and Department of Computer Science, Cornell University
Ithaca, NY 14853.
`shmoys@cs.cornell.edu`

³ Department of Computer Science, Cornell University, Ithaca, NY 14853.
`swamy@cs.cornell.edu`

1 Introduction

There has been a great deal of recent work on approximation algorithms for facility location problems [9]. We consider the *capacitated facility location problem with hard capacities*. We are given a set of facilities, \mathcal{F} , and a set of clients \mathcal{D} in a common metric space. Each facility i has a *facility opening cost* f_i and *capacity* u_i that specifies the maximum number of clients that may be assigned to this facility. We want to *open* some facilities from the set \mathcal{F} and assign each client to an open facility so that at most u_i clients are assigned to any open facility i . The cost of assigning client j to facility i is given by their distance c_{ij} , and our goal is to minimize the sum of the facility opening costs and the client assignment costs.

The recent work on facility location problems has come in two varieties: LP-based algorithms, and local search-based algorithms. For the problem described above, no constant approximation algorithm based on LP is known, and in fact, no LP relaxation is known for which the ratio between the optimal integer and fractional values has been bounded by a constant. Surprisingly, constant performance guarantees can still be proved based on local search, but these have the disadvantage that on an instance by instance basis, one never knows anything stronger than the guarantee itself (without resorting to solving an LP relaxation anyway).

We present an algorithm that rounds the optimal fractional solution to a natural LP relaxation by using this solution to guide the decomposition of the input into a collection of single-demand-node capacitated facility location problems, which are then solved independently. In the special case that *all facility opening costs are equal*, we show that our algorithm is a 5-approximation algorithm, thereby also providing the first constant upper bound on the integrality gap of this formulation in this important special case. One salient feature of our

* Research supported partially by a grant from Motorola and NSF grant CCR-9912422.

** Research supported partially by NSF grant CCR-9912422.

algorithm is that it relies on a decomposition of the input into instances of the single-demand capacitated facility location problem; in this way, the algorithm mirrors the work of Aardal [1], who presents a computational polyhedral approach for this problem which uses the same core problem in the identification of cutting planes.

There are several variants of the capacitated facility location problem, which have rather different properties, especially in terms of the approximation algorithms that are currently known. One distinction is between *soft* and *hard* capacities: in the latter problem, each facility is either opened at some location or not, whereas in the former, one may specify any integer number of facilities to be opened at that location. Soft capacities make the problem easier; Shmoys, Tardos, & Aardal [11] gave the first constant approximation algorithm for this problem based on an LP-rounding technique; Jain & Vazirani [4] gave a general technique for converting approximation algorithm results for the uncapacitated problem into algorithms that can handle soft capacities. Korupolu, Plaxton, & Rajaraman [5] gave the first constant approximation algorithm that handles hard capacities, based on a local search procedure, but their approach worked only if all capacities are equal. Chudak & Williamson [3] improved this performance guarantee to 5.83 for the same uniform capacity case. Pál, Tardos, & Wexler [8] gave the first constant performance guarantee for the case of non-uniform hard capacities. This was recently improved by Mahdian & Pál [6] and Zhang, Chen, & Ye [13] to yield a 5.83-approximation algorithm.

There is also a distinction between the case of *unsplittable assignments* and *splittable* ones. That is, suppose that each client j has a certain demand d_j to be assigned to open facilities so that the total demand assigned to each facility is at most its capacity: does each client need to have all of its demand served by a unique facility? In the former case, the answer is yes, whereas in the latter, the answer is no. All approximation algorithms for hard capacities have focused on the splittable case. Note that once one has decided which facilities to open, the optimal splittable assignment can be computed by solving a transportation problem. A splittable assignment can be converted to an unsplittable one at the cost of increasing the required capacity at each facility (using an approximation algorithm for the generalized assignment problem [10]). Of course, if there are integer capacities and all demands are 1, there is no distinction between the two problems.

For hard capacities, it is easy to show that the natural LP formulations do not have any constant integrality ratio; the simplest such example has two facility locations, one essentially free, and one very expensive. In contrast, we focus on the case in which all facility opening costs are equal. For ease of exposition, we will focus on the case in which each demand is equal to 1. However, it is a relatively straightforward exercise to extend the algorithm and its analysis to the case of general demands. We will use the terms “assignment cost” and “service cost” interchangeably.

Our Techniques. The outline of our algorithm is as follows. Given the optimal LP solution and its dual, we view the optimal primal solution as a bipartite graph in which the nodes correspond to facility locations and clients, and the edges correspond to pairs (i, j) such that a positive fraction of the demand at client j is assigned to facility i by the LP solution. We use this to construct a partition of the demand and facilities into clusters: each cluster is “centered” at a client, and the neighbors of this client contained in the cluster are opened (in the fractional solution) in total at least $1/2$. Each fractionally open facility location will, ultimately, be assigned to some cluster (i.e., not every facility assigned to this cluster need be a neighbor of the center), and each cluster will be expected to serve all of the demand that its facilities serve in the fractional solution. Each facility i that is fully opened in the fractional solution can immediately be opened and serve all of its demand; we view the remaining demand as located at the cluster center, and find a solution to the single-demand capacitated facility location problem induced by this cluster to determine the other facilities to open within this cluster. Piecing this together for each cluster, we then solve a transportation problem to determine the corresponding assignment.

To analyze this procedure, we show that the LP solution can also be decomposed into feasible fractional solutions to the respective single-demand problems. Our algorithm for the single-node subproblems computes a rounding of this fractional solution, and it is important that we can bound the increase in cost incurred by this rounding. Furthermore, note that it will be important for the analysis (and the effectiveness of the algorithm) that we ensure that in moving demand to a cluster center, we are not moving it too much, since otherwise the solution created for the single-node problem will be prohibitively expensive for the true location of the demand.

One novel aspect of our analysis is that the performance guarantee analysis comes in two parts: a part that is related to the fact that the assignment costs are increased by this displacement of the demand, and a part that is due to the aggregated effect of rounding the fractional solutions to the single-node problems. One consequence of this is that our analysis is not the “client-by-client” analysis that has become the dominant paradigm in the recent flurry of work in this area. Finally, our analysis relies on both the primal and dual LPs to bound the cost of the solution computed. In doing this, one significant difficulty is that the terms in the dual objective that correspond to the upper bound for the hard capacity have a -1 as their coefficient; however, we show that further structure in the optimal primal-dual pair that results from the complementary slackness conditions is sufficient to overcome this obstacle (in a way similar to that used earlier in [12]).

Although our analysis applies only to the case in which the fixed costs are equal, our algorithm is sufficiently general to handle arbitrary fixed costs. Furthermore, we believe that our approach may prove to be a useful first step in analyzing more sophisticated LP relaxations of the capacitated facility location problem; in particular, we believe that the decomposition into single-node problems can be a provable effective approach in the more general case. Specifically,

we conjecture that the extended flow cover inequalities of Padberg, Van Roy, and Wolsey [7] as adapted by Aardal [1] are sufficient to insure a constant integrality gap; this raises the possibility of building on a recent result of Carr, Fleischer, Leung, and Phillips [2] that showed an analogous result for the single-demand node problem.

2 A Linear Program

We can formulate the capacitated facility location problem as an integer program and relax the integrality constraints to get a linear program (LP). We use i to index the facilities in \mathcal{F} and j to index the clients in \mathcal{D} .

$$\min \quad \sum_i f_i y_i + \sum_j \sum_i d_j c_{ij} x_{ij} \quad (\text{P})$$

$$\text{s.t.} \quad \sum_i x_{ij} \geq 1 \quad \forall j \quad (1)$$

$$x_{ij} \leq y_i \quad \forall i, j \quad (2)$$

$$\sum_j d_j x_{ij} \leq u_i y_i \quad \forall i \quad (3)$$

$$y_i \leq 1 \quad \forall i \quad (4)$$

$$x_{ij}, y_i \geq 0 \quad \forall i, j.$$

Variable y_i indicates if facility i is open and x_{ij} indicates the fraction of the demand of client j that is assigned to facility i . The first constraint states that each client must be assigned to a facility. The second constraint says that if client j is assigned to facility i then i must be open, and constraint (3) says that at most u_i amount of demand may be assigned to i . Finally (4) says that a facility can only be opened once. A solution where the y_i variables are 0 or 1 corresponds exactly to a solution to our problem. The dual program is,

$$\max \quad \sum_j \alpha_j - \sum_i z_i \quad (\text{D})$$

$$\text{s.t.} \quad \alpha_j \leq d_j c_{ij} + \beta_{ij} + d_j \gamma_i \quad \forall i, j \quad (5)$$

$$\sum_j \beta_{ij} \leq f_i + z_i - u_i \gamma_i \quad \forall i \quad (6)$$

$$\alpha_j, \beta_{ij}, \gamma_i, z_i \geq 0 \quad \forall i, j.$$

Intuitively α_j is the *budget* that j is willing to spend to get itself assigned to an open facility. Constraint (5) says that a part of this is used to pay for the assignment cost $d_j c_{ij}$ and the rest is used to (partially) pay for the facility opening cost.

For convenience, in what follows, we consider unit demands, i.e., $d_j = 1$ for all j . The primal constraint (3) and the dual constraint (5) then simplify to,

$\sum_j x_{ij} \leq u_i y_i$, and $\alpha_j \leq c_{ij} + \beta_{ij} + \gamma_i$, and the objective function of the primal program (P) is $\min \sum_i f_i y_i + \sum_{j,i} c_{ij} x_{ij}$. All our results continue to hold in the presence of arbitrary demands d_j if the demand of a client is allowed to be assigned to multiple facilities.

3 Rounding the LP

In this section we give a 5-approximation algorithm for capacitated facility location when all facility costs are equal. We will round the optimal solution to (P) to an integer solution losing a factor of at most 5, thus obtaining a 5-approximation algorithm.

3.1 The Single-Demand-Node Capacitated Facility Location Problem

The special case of capacitated facility location where we have just one client or demand node (called SNCFL) plays an important role in our rounding algorithm. This is also known as the single-node fixed-charge problem [7] or the single-node capacitated flow problem. The linear program (P) simplifies to the following.

$$\min \quad \sum_i f_i v_i + \sum_i c_i w_i \quad (\text{SN-P})$$

$$\text{s.t.} \quad \sum_i w_i \geq D$$

$$w_i \leq u_i v_i \quad \forall i \quad (7)$$

$$v_i \leq 1 \quad \forall i \quad (8)$$

$$w_i, v_i \geq 0 \quad \forall i.$$

Here D is the total demand that has to be assigned, $f_i \geq 0$ is the fixed cost of facility i , and $c_i \geq 0$ is the per unit cost of sending flow, or distance, to facility i . Variable w_i is the *total demand* (or *flow*) assigned to facility i , and v_i indicates if facility i is open. We show that a simple greedy algorithm returns an optimal solution to (SN-P) that has the property that at most one facility is fractionally open, i.e., there is at most one i such that $0 < v_i < 1$. We will exploit this fact in our rounding scheme.

Given any feasible solution (w, v) we can set $\hat{v}_i = \frac{w_i}{u_i}$ and obtain a feasible solution (w, \hat{v}) of no greater cost. So we can eliminate the v_i variables from (SN-P), changing the objective function to $\min \sum_i (\frac{f_i}{u_i} + c_i) w_i$, and replacing constraints (7), (8) by $w_i \leq u_i$ for each i . Clearly, this is equivalent to the earlier formulation. It is easy to see now that the following greedy algorithm delivers an optimal solution: start with $w_i = v_i = 0$ for all i . Consider facilities in increasing order of $\frac{f_i}{u_i} + c_i$ value and assign to facility i a demand equal to u_i or the residual demand left, whichever is smaller, i.e., set $w_i = \min(u_i, \text{demand left})$, $v_i = \frac{w_i}{u_i}$, until all D units of demand have been assigned. We get the following lemma.

Lemma 3.1. *The greedy algorithm that assigns demand to facilities in increasing order of $\frac{f_i}{u_i} + c_i$ delivers an optimal solution to (SN-P). Furthermore, there is at most one facility i in the optimal solution such that $0 < v_i < 1$.*

3.2 The Algorithm

We now describe the full rounding procedure. Let (x, y) and $(\alpha, \beta, \gamma, z)$ be the optimal solutions to (P) and (D) respectively, and OPT be the common optimal value. We may assume without loss of generality that $\sum_i x_{ij} = 1$ for every client j . We first give an overview of the algorithm.

Our algorithm runs in two phases. In the first phase, we partition the facilities i such that $y_i > 0$ into *clusters* each of which will be “centered” around a client that we will call the *cluster center*. We denote the cluster centered around client k by N_k . The cluster N_k consists of its center k , the set of facilities assigned to it, and the fractional demand served by these facilities, i.e., $\sum_{i \in N_k} \sum_j x_{ij}$. The clustering phase maintains two properties that will be essential for the analysis. It ensures that, (1) each cluster contains a fractional facility weight of at least $\frac{1}{2}$, i.e., $\sum_{i \in N_k} y_i \geq \frac{1}{2}$, and (2) if some facility in cluster N_k fractionally serves a client j , then the center k is not “too far” away from j (we make this precise in the analysis). To maintain the second property we require a somewhat more involved clustering procedure than the one in [11]. In the second phase of the algorithm we decide which facilities will be (fully) opened in each cluster. We consider each cluster separately, and open enough facilities in N_k to serve the fractional demand associated with the cluster. This is done in two steps. First, we open every facility in N_k with $y_i = 1$. Next, we set up an instance of SNCFL. The instance consists of all the remaining facilities, and the entire demand served by these facilities, $D_k = \sum_{i \in N_k; y_i < 1} \sum_j x_{ij}$, considered as concentrated at the center k . Now we use the greedy algorithm above to obtain an optimal solution to this instance with the property that at most one facility is fractionally open. Since the facility costs are all equal and each cluster has enough facility weight, we can fully open this facility and charge this against the cost that the LP incurs in opening facilities from N_k . Piecing together the solutions for the different clusters gives a solution to the capacitated facility location instance, in which every facility is either fully open or closed. Now we compute the min-cost assignment of clients to open facilities by solving a transportation problem.

We now describe the algorithm in detail. Let $F = \{i : y_i > 0\}$ be the (partially) opened facilities in (x, y) , and $F_j = \{i : x_{ij} > 0\}$ be the facilities in F that fractionally serve client j .

1. **Clustering.** This is done in two steps.

- C1. At any stage, let \mathcal{C} be the set of the current cluster centers, which is initially empty. We use N_k to denote a cluster centered around client $k \in \mathcal{C}$. For each client $j \notin \mathcal{C}$, we maintain a set B_j of unclustered facilities that are closer to it than to any cluster center, i.e., $B_j = \{i \in F_j : i \notin \bigcup_{k \in \mathcal{C}} N_k \text{ and } c_{ij} \leq \min_{k \in \mathcal{C}} c_{ik}\}$. (This definition of B_j is crucial in our analysis that shows that if client j is fractionally served by N_k , then

k is not “too far” from j .) We also have a set \mathcal{S} containing all clients that could be chosen as cluster centers. These are all clients $j \notin \mathcal{C}$ that send at least half of their demand to facilities in B_j , i.e., $\mathcal{S} = \{j \notin \mathcal{C} : \sum_{i \in B_j} x_{ij} \geq \frac{1}{2}\}$. Of course, initially $\mathcal{S} = \mathcal{D}$, since $\mathcal{C} = \emptyset$.

While \mathcal{S} is not empty, we repeatedly pick $j \in \mathcal{S}$ with smallest α_j value and form the cluster $N_j = B_j$ around it. We update the sets \mathcal{C} and \mathcal{S} accordingly. (Note that for any cluster N_k , we have that $\sum_{i \in N_k} y_i \geq \sum_{i \in N_k} x_{ik} \geq \frac{1}{2}$.)

- C2. After the previous step, there could still be facilities that are not assigned to any cluster. We now assign these facilities in $U = F - \bigcup_{k \in \mathcal{C}} N_k$ to clusters. We assign $i \in U$ to the cluster whose center is nearest to it, i.e., we set $N_j = N_j \cup \{i\}$ where $j = \operatorname{argmin}_{k \in \mathcal{C}} c_{ik}$. In addition, we assign to the cluster all of the fractional demand served by facility i . (After this step, the clusters $N_j, j \in \mathcal{C}$ partition the set of facilities F .)
- 2. **Reducing to the single-node instances.** For each cluster N_k , we first open each facility i in N_k with $y_i = 1$. We now create an instance of SNCFL on the *remaining* set of facilities, by considering the total demand assigned to these facilities as being concentrated at the cluster center k . So our set of facilities is $L_k = \{i \in N_k : y_i < 1\}$, each c_i is the distance c_{ik} , and the total demand is $D_k = \sum_{i \in L_k} \sum_j x_{ij}$. We use the greedy algorithm of Section 3.1 to find an optimal solution $(w^{(k)}, v^{(k)})$ to this linear program. Let O_k^* be the value of this solution. We call the facility i such that $0 < w_i^{(k)} < 1$ (if such a facility exists) the *extra facility* in cluster N_k . We fully open all the facilities in L_k with $w_i^{(k)} > 0$ (including the extra facility). Note that the facilities opened (including each i such that $y_i = 1$) have enough capacity to satisfy all the demand $\sum_{i \in N_k} \sum_j x_{ij}$. Piecing together the solutions for all the clusters, we get a solution where all the y variables are assigned values in $\{0, 1\}$.
- 3. **Assigning clients.** We compute a minimum cost assignment of clients to open facilities by solving the corresponding transportation problem.

3.3 Analysis

The performance guarantee of our algorithm will follow from the fact that the decomposition constructed by the algorithm of the original problem instance into single-node subproblems, one for each cluster, satisfies the following two nice properties. First, in Lemma 3.5, we show that the total cost of the optimal solutions for each of these single-node instances is not too large compared to OPT . We prove this by showing that the LP solution induces a feasible solution to (SN-P) for the SNCFL instance of each cluster and that the total cost of these feasible solutions is bounded. Second, in Lemma 3.7, we show that the optimal solutions to each of these single-node instances obtained by our greedy algorithm in Section 3.1, can be mapped back to yield a solution to the original problem in which every facility is either opened fully, or not opened at all, while losing a small additive term. Piecing together these partial solutions, we construct a

solution to the capacitated facility location problem. The cost of this solution is bounded by aggregating the bounds obtained for each partial solution. We note that this bound is not based on a “client-by-client” analysis, but rather on bounding the cost generated by the overall cluster.

Observe that there are two sources for the extra cost involved in mapping the solutions to the single-node instances. We might need to open one fractionally open facility in the optimal fractional solution to (SN-P). This is bounded in Lemma 3.6, and this is the only place in the entire proof which uses the assumption that the fixed costs are all equal. In addition, we need to transfer all of the fractional demand that was assumed to be concentrated at the center of the cluster, back to its original location. To bound the extra assignment cost involved, we rely on the important fact that if a client j is fractionally served by some facility $i \in N_k$, then the distance c_{jk} is bounded. Since the triangle inequality implies that $c_{jk} \leq c_{ij} + c_{ik}$, we focus on bounding the distance c_{ik} . This is done in Lemmas 3.3 and 3.4. In Lemma 3.8, we provide a bound on the facility cost and assignment cost involved in opening the facilities with $y_i = 1$, which, by relying on complementary slackness, overcomes the difficulties posed by the $-z_i$ term in the dual objective function.

We then combine these bounds to prove our main theorem, Theorem 3.9, which states that the resulting feasible solution for the capacitated facility location problem is of cost at most $5 \cdot OPT$.

We first prove the following lemma that states a necessary condition for a facility i to be assigned to cluster N_k .

Lemma 3.2. *Let i be a facility assigned to cluster N_k in step C1 or C2. Let \mathcal{C}' be the set of cluster centers just after this assignment. Then, k is the cluster center closest to i among all cluster centers in \mathcal{C}' ; that is, $c_{ik} = \min_{k' \in \mathcal{C}'} c_{ik'}$.*

Proof. Since $k \in \mathcal{C}'$, clearly we have that $c_{ik} \geq \min_{k' \in \mathcal{C}'} c_{ik'}$. If i is assigned in step C1, then it must be included when the cluster centered at k is first formed; that is, $i \in B_k$ and the lemma holds by the definition of B_k . Otherwise, if i is assigned in step C2, then \mathcal{C}' is the set of all cluster centers, in which case it is again true by definition. \square

For a client j , consider the point when j was removed from the set \mathcal{S} in step C1, either because a cluster was created around it, or because the weight of the facilities in B_j decreased below $\frac{1}{2}$ when some other cluster was created. Let $A_j = F_j \setminus B_j$ be the set of facilities not in B_j at that point. Recall that there are two reasons for removing a facility i from the set B_j : it was assigned to some cluster N_k , or there was some cluster center $k' \in \mathcal{C}$, such that $c_{ik'} < c_{ij}$. We define $i^*(j)$ as the facility in A_j nearest to j . Also, observe that once $j \notin \mathcal{C} \cup \mathcal{S}$, then we have that $\sum_{i \in A_j} x_{ij} > \frac{1}{2}$.

Lemma 3.3. *Consider any client j and any facility $i \in A_j$. If i is assigned to cluster N_k , then $c_{ik} \leq \alpha_j$.*

Proof. If $k = j$, (j could be a cluster center), then we are done since $A_j \subseteq F_j$ and $x_{ij} > 0$ implies that $c_{ij} \leq \alpha_j$ (by complementary slackness). Otherwise,

consider the point when j was removed from \mathcal{S} in step C1, and let \mathcal{C}' be the set of cluster centers just after j is removed. Note that j could belong to \mathcal{C}' if it is a cluster center. Since $i \notin B_j$ at this point, either $i \in N_{k'}$ for some $k' \in \mathcal{C}'$ or we have that $c_{ij} > \min_{k' \in \mathcal{C}' - \{j\}} c_{ik'}$. In the former case, it must be that $k' = k$, since the clusters are disjoint. Also, $c_{ik} \leq \alpha_k$, since $N_k \subseteq F_k$, and $\alpha_k \leq \alpha_j$, since k was picked before j from \mathcal{S} (recall the order in which we consider clients in \mathcal{S}). In the latter case, consider the set of cluster centers \mathcal{C}'' just after i is assigned to N_k (either in step C1 or step C2), and so $k \in \mathcal{C}''$. It must be that $\mathcal{C}'' \supseteq \mathcal{C}'$, since i was removed from B_j before it was assigned to N_k , and by Lemma 3.2, $c_{ik} = \min_{k' \in \mathcal{C}''} c_{ik'}$. Hence, $c_{ik} \leq \min_{k' \in \mathcal{C}' - \{j\}} c_{ik'} < c_{ij} \leq \alpha_j$ since $A_j \subseteq F_j$. \square

Lemma 3.4. *Consider any client j and a facility $i \in F_j \setminus A_j$. Let i be assigned to cluster N_k . If $j \in \mathcal{C}$, then $c_{ik} \leq c_{ij}$; otherwise, $c_{ik} \leq c_{ij} + c_{i^*(j)j} + \alpha_j$.*

Proof. If j is a cluster center, then when it was removed from \mathcal{S} , we have constructed the cluster N_j equal to the current set B_j , which is precisely $F_j \setminus A_j$. So i is assigned to N_j , that is, $k = j$, and hence the bound holds.

Suppose $j \notin \mathcal{C}$. Consider the point just before the facility $i^*(j)$ is removed from the set B_j in step C1, and let \mathcal{C}' be the set of cluster centers at this point. By the definition of the set A_j , j is still a candidate cluster center at this point. Let $k' \in \mathcal{C}'$ be the cluster center due to which $i^*(j)$ was removed from B_j , and so either $i^*(j) \in N_{k'} \subseteq F_{k'}$ or $c_{i^*(j)k'} < c_{i^*(j)j}$. In each case, we have $c_{i^*(j)k'} \leq \alpha_j$, since the choice of k' implies that $\alpha_{k'} \leq \alpha_j$. Now consider the set of cluster centers \mathcal{C}'' just after i is assigned to N_k . Since $i \notin A_j$, $i^*(j)$ was removed from B_j before this point. So we have $\mathcal{C}'' \supseteq \mathcal{C}'$. Using Lemma 3.2,

$$c_{ik} = \min_{k'' \in \mathcal{C}''} c_{ik''} \leq c_{ik'} \leq c_{ij} + c_{i^*(j)j} + c_{i^*(j)k'} \leq c_{ij} + c_{i^*(j)j} + \alpha_j.$$

□

Consider now any cluster N_k . Recall that $L_k = \{i \in N_k : y_i < 1\}$, $(w^{(k)}, v^{(k)})$ is the optimal solution to (SN-P) found by the greedy algorithm for the single-node instance corresponding to this cluster, and O_k^* is the value of this solution. Let $k(i) \in \mathcal{C}$ denote the cluster to which facility i is assigned, and so $i \in N_{k(i)}$.

Lemma 3.5. *The optimal value $O_k^* \leq \sum_{i \in L_k} f_i y_i + \sum_{j,i \in L_k} c_{ik} x_{ij}$, and hence, $\sum_{k \in \mathcal{C}} O_k^* \leq \sum_{i:y_i < 1} f_i y_i + \sum_{j,i:y_i < 1} c_{ik(i)} x_{ij}$.*

Proof. The second bound follows from the first since the clusters N_k are disjoint. We will upper bound O_k^* by exhibiting a feasible solution (\hat{w}, \hat{v}) of cost at most the claimed value. Set $\hat{v}_i = y_i$, and $\hat{w}_i = \sum_j x_{ij}$ for all $i \in L_k$. Note that $\sum_i \hat{w}_i = \sum_{i \in L_k} \sum_j x_{ij} = D_k$. The facility cost of this solution is at most $\sum_{i \in L_k} f_i \hat{v}_i = \sum_{i \in L_k} f_i y_i$. The service cost is $\sum_{i \in L_k} c_i \hat{w}_i = \sum_{j,i \in L_k} c_{ik} x_{ij}$. Combining this with the bound on facility cost proves the lemma. \square

Lemma 3.6. *The cost of opening the (at most one) extra facility in cluster N_k is at most $2 \sum_{i \in N_k} f_i y_i$.*

Proof. We have $\sum_{i \in N_k} y_i \geq \sum_{i \in N_k} x_{ik} \geq \frac{1}{2}$ since N_k was created in step C1 and is centered around k , and no facility is removed from N_k in step C2. We open at most one extra facility from N_k . Since all facilities have the same cost f , the cost of opening this facility is $f \leq f \cdot 2 \sum_{i \in N_k} y_i = 2 \sum_{i \in N_k} f_i y_i$. This is the only place where we use the fact that the facility costs are all equal. \square

Let \hat{y} be the 0-1 vector indicating which facilities are open, i.e., $\hat{y}_i = 1$ if i is open, and 0 otherwise. We let $\hat{y}^{(k)}$ denote the portion of \hat{y} consisting of the facilities in L_k , i.e., $\hat{y}^{(k)} = (\hat{y}_i^{(k)})_{i \in L_k}$ and $\hat{y}_i^{(k)} = 1$ if $i \in L_k$ is open, and 0 otherwise.

Lemma 3.7. *The solution $(w^{(k)}, v^{(k)})$ for cluster N_k yields an assignment $\hat{x}^{(k)} = (\hat{x}_{ij}^{(k)})_{i \in L_k, j \in \mathcal{D}}$ such that,*

- (i) $(\hat{x}^{(k)}, \hat{y}^{(k)})$ obeys constraints (2)–(4) for all $i \in L_k$,
- (ii) \hat{x} satisfies $\sum_{i \in L_k} x_{ij}$ fraction of the demand of each client j , that is, $\sum_{i \in L_k} \hat{x}_{ij} = \sum_{i \in L_k} x_{ij}$ for all j and,
- (iii) the cost $\sum_{i \in L_k} f_i \hat{y}_i^{(k)} + \sum_{j, i \in L_k} c_{ij} \hat{x}_{ij}^{(k)}$ is at most $O_k^* + 2 \sum_{i \in N_k} f_i y_i + \sum_{j, i \in L_k} c_{ij} x_{ij} + \sum_{j, i \in L_k} c_{ik} x_{ij}$.

Proof. We have $O_k^* = \sum_{i \in L_k} (f_i v_i^{(k)} + c_i w_i^{(k)})$. Constraints (4) are clearly satisfied for $i \in L_k$, since $\hat{y}^{(k)}$ is a $\{0, 1\}$ -vector. The facility cost $\sum_{i \in L_k} f_i \hat{y}_i^{(k)}$ is at most $\sum_{i \in L_k} f_i v_i^{(k)} + 2 \sum_{i \in N_k} f_i y_i$ since every facility other than the extra facility is either fully open or not open in the solution $(w^{(k)}, v^{(k)})$ and the cost of opening the extra facility is at most $2 \sum_{i \in N_k} f_i y_i$ by Lemma 3.6.

We set the variables $\hat{x}_{ij}^{(k)}$ for $i \in L_k$ so that the service cost $\sum_{j, i \in L_k} c_{ij} \hat{x}_{ij}^{(k)}$ can be bounded by $\sum_{i \in L_k} c_i w_i^{(k)} + \sum_{j, i \in L_k} (c_{ij} + c_{ik}) x_{ij}$. Combining this with the above bound on the facility cost, proves the lemma. The service cost of the single-node solution is the cost of transporting the entire demand $D_k = \sum_{j, i \in L_k} x_{ij}$ from the facilities in L_k to the center k , and now we want to move the demand, $\sum_{i \in L_k} x_{ij}$, of client j from k back to j . Doing this for every client j incurs an additional cost of $\sum_j \sum_{i \in L_k} c_{jk} x_{ij} \leq \sum_{j, i \in L_k} (c_{ij} + c_{ik}) x_{ij}$. More precisely, we set $\hat{x}_{ij}^{(k)}$, $i \in L_k$ arbitrarily so that, (1) $\sum_{i \in L_k} \hat{x}_{ij}^{(k)} = \sum_{i \in L_k} x_{ij}$ for every client j , and (2) $\sum_j \hat{x}_{ij}^{(k)} = w_i^{(k)}$ for every facility $i \in L_k$. This satisfies constraints (2), (3) — if $\hat{x}_{ij}^{(k)} > 0$ then $w_i^{(k)} > 0$, so $\hat{y}_i^{(k)} = 1$, and $\sum_j \hat{x}_{ij}^{(k)} = w_i^{(k)} \leq u_i = u_i \hat{y}_i^{(k)}$. The service cost is,

$$\sum_{j, i \in L_k} c_{ij} \hat{x}_{ij}^{(k)} \leq \sum_{i \in L_k, j} c_{ik} \hat{x}_{ij}^{(k)} + \sum_{j, i \in L_k} c_{jk} \hat{x}_{ij}^{(k)} \leq \sum_{i \in L_k} c_i w_i^{(k)} + \sum_{j, i \in L_k} (c_{ij} + c_{ik}) x_{ij}.$$

\square

Lemma 3.8. *The cost of opening facilities i with $y_i = 1$, and for each such i , of sending x_{ij} units of flow from j to i for every client j , is at most $\sum_{j,i:y_i=1} \alpha_j x_{ij} - \sum_i z_i$.*

Proof. This follows from complementary slackness. Each facility i with $z_i > 0$ has $y_i = 1$. For any such facility we have,

$$\begin{aligned} \sum_j \alpha_j x_{ij} &= \sum_j c_{ij} x_{ij} + \sum_j \beta_{ij} x_{ij} + \sum_j \gamma_i x_{ij} \quad (x_{ij} > 0 \Rightarrow \alpha_j = c_{ij} + \beta_{ij} + \gamma_i) \\ &= \sum_j c_{ij} x_{ij} + \sum_j \beta_{ij} y_i + u_i \gamma_i y_i \quad \begin{cases} \beta_{ij} > 0 \Rightarrow x_{ij} = y_i, \\ \gamma_i > 0 \Rightarrow \sum_j x_{ij} = u_i y_i \end{cases} \\ &= \sum_j c_{ij} x_{ij} + f_i + z_i. \quad \begin{cases} y_i > 0 \Rightarrow \sum_j \beta_{ij} + u_i \gamma_i \\ = f_i + z_i \end{cases} \end{aligned}$$

Summing over all i with $y_i = 1$ proves the lemma. \square

Putting the various pieces together, we get the following theorem.

Theorem 3.9. *The cost of the solution returned is at most $5 \cdot OPT$.*

Proof. To bound the total cost, it suffices to give a fractional assignment (\hat{x}_{ij}) such that (\hat{x}, \hat{y}) is a feasible solution to (P) and has cost at most $5 \cdot OPT$. We construct the fractional assignment as follows. First we set $\hat{x}_{ij} = x_{ij}$ for every facility i with $y_i = 1 = \hat{y}_i$. This satisfies constraints (2)–(4) for i such that $y_i = 1$. By the previous lemma we have,

$$\sum_{i:y_i=1} f_i \hat{y}_i + \sum_{j,i:y_i=1} c_{ij} \hat{x}_{ij} = \sum_{j,i:y_i=1} \alpha_j x_{ij} - \sum_i z_i. \quad (9)$$

Now for each cluster N_k , we set $\hat{x}_{ij} = \hat{x}_{ij}^{(k)}$ for $i \in L_k$ where $(\hat{x}^{(k)}, \hat{y}^{(k)})$ is the partial solution for cluster N_k given by Lemma 3.7. All other \hat{x}_{ij} variables are 0. Applying parts (i) and (ii) of Lemma 3.7 for all $k \in \mathcal{C}$, we get that (\hat{x}, \hat{y}) satisfies (2)–(4) for every i such that $y_i < 1$, and $\sum_{i:y_i<1} \hat{x}_{ij} = \sum_{i:y_i<1} x_{ij}$ for every client j . Hence, (\hat{x}, \hat{y}) satisfies constraints (2)–(4) and $\sum_i \hat{x}_{ij} = \sum_{i:y_i=1} x_{ij} + \sum_{i:y_i<1} x_{ij} = 1$, showing that (\hat{x}, \hat{y}) is a feasible solution to (P). Since the clusters N_k are disjoint, from part (iii) of Lemma 3.7, we have,

$$\begin{aligned} \sum_{i:y_i<1} f_i \hat{y}_i + \sum_{j,i:y_i<1} c_{ij} \hat{x}_{ij} &\leq \sum_{k \in \mathcal{C}} O_k^* + 2 \sum_i f_i y_i + \sum_{j,i:y_i<1} c_{ij} x_{ij} + \sum_{j,i:y_i<1} c_{ik(i)} x_{ij} \\ &\leq 3 \sum_i f_i y_i + \sum_{j,i:y_i<1} c_{ij} x_{ij} + 2 \sum_{j,i:y_i<1} c_{ik(i)} x_{ij}. \end{aligned}$$

where the last inequality follows from Lemma 3.5. For any client j and facility $i \in F_j$, if $i \in A_j$, then we have $c_{ik(i)} \leq \alpha_j$ by Lemma 3.3; otherwise, by Lemma 3.4,

$c_{ik(i)} \leq c_{ij} \leq c_{ij} + \alpha_j$ for $j \in \mathcal{C}$, and $c_{ik(i)} \leq c_{ij} + c_{i^*(j)j} + \alpha_j$ for $j \notin \mathcal{C}$. Plugging this in the above expression we get,

$$\begin{aligned} \sum_{i:y_i < 1} f_i \hat{y}_i + \sum_{j,i:y_i < 1} c_{ij} \hat{x}_{ij} &\leq 3 \sum_i f_i y_i + \sum_{j,i:y_i < 1} c_{ij} x_{ij} + 2 \sum_{j,i:y_i < 1} \alpha_j x_{ij} \\ &\quad + 2 \sum_j \sum_{\substack{i:y_i < 1 \\ i \notin A_j}} c_{ij} x_{ij} + \sum_{j \notin \mathcal{C}} 2c_{i^*(j)j} \sum_{\substack{i:y_i < 1 \\ i \notin A_j}} x_{ij}. \end{aligned}$$

For $j \notin \mathcal{C}$, $\sum_{i \notin A_j} x_{ij} < \frac{1}{2}$. So $2c_{i^*(j)j} \left(\sum_{i:y_i < 1, i \notin A_j} x_{ij} \right)$ is at most,

$$c_{i^*(j)j} = \min_{i \in A_j} c_{ij} \leq \frac{\sum_{i \in A_j} c_{ij} x_{ij}}{\sum_{i \in A_j} x_{ij}} < 2 \sum_{i \in A_j} c_{ij} x_{ij}.$$

This implies that,

$$\begin{aligned} \sum_{i:y_i < 1} f_i \hat{y}_i + \sum_{j,i:y_i < 1} c_{ij} \hat{x}_{ij} &\leq 3 \sum_i f_i y_i + \sum_{j,i:y_i < 1} c_{ij} x_{ij} + 2 \sum_{j,i:y_i < 1} \alpha_j x_{ij} \\ &\quad + 2 \sum_j \sum_{\substack{i:y_i < 1 \\ i \notin A_j}} c_{ij} x_{ij} + 2 \sum_{j \notin \mathcal{C}} \sum_{i \in A_j} c_{ij} x_{ij} \\ &\leq 2 \sum_{j,i:y_i < 1} \alpha_j x_{ij} + 3 \left(\sum_i f_i y_i + \sum_{j,i} c_{ij} x_{ij} \right). \end{aligned} \quad (10)$$

Finally, combining (9) and (10), we obtain that

$$\begin{aligned} \text{Total Cost} &\leq \left(\sum_{j,i:y_i=1} \alpha_j x_{ij} - \sum_i z_i \right) + 2 \sum_{j,i:y_i < 1} \alpha_j x_{ij} + 3 \left(\sum_i f_i y_i + \sum_{j,i} c_{ij} x_{ij} \right) \\ &\leq 2 \left(\sum_{j,i:y_i=1} \alpha_j x_{ij} - \sum_i z_i + \sum_{j,i:y_i < 1} \alpha_j x_{ij} \right) + 3 \cdot OPT = 5 \cdot OPT. \end{aligned}$$

□

References

- [1] K. Aardal. Capacitated facility location: separation algorithms and computational experience. *Mathematical Programming*, 81:149–175, 1998.
- [2] R. Carr, L. Fleischer, V. Leung, and C. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 106–115, 2000.
- [3] F. A. Chudak and D. P. Williamson. Improved approximation algorithms for capacitated facility location problems. In G. Cornuéjols, R. E. Burkard, and G. J. Woeginger, editors, *Integer Programming and Combinatorial Optimization*, volume 1610 of *Lecture Notes in Computer Science*, pages 99–113, Graz, 1999. Springer.

- [4] K. Jain and V.V. Vazirani. Approximation algorithms for metric facility location and k -median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48:274–296, 2001.
- [5] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37(1):146–188, 2000.
- [6] M. Mahdian and M. Pál. Universal facility location. In *Proceedings of the 11th ESA*, pages 409–421, 2003.
- [7] M. W. Padberg, T. J. Van Roy, and L. A. Wolsey. Valid linear inequalities for fixed charge problems. *Operations Research*, 33:842–861, 1985.
- [8] M. Pál, É. Tardos, and T. Wexler. Facility location with nonuniform hard capacities. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 329–338, 2001.
- [9] D. B. Shmoys. Approximation algorithms for facility location problems. In *Proceedings of 3rd APPROX*, pages 27–33, 2000.
- [10] D. B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming A*, 62:461–474, 1993.
- [11] D. B. Shmoys, É. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.
- [12] C. Swamy and D. B. Shmoys. Fault-tolerant facility location. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 735–736, 2003.
- [13] J. Zhang, B. Chen, and Y. Ye. A multi-exchange local search algorithm for the capacitated facility location problem. Submitted to *Mathematics of Operations Research*, 2003.

A Multi-exchange Local Search Algorithm for the Capacitated Facility Location Problem^{*} (Extended Abstract)

Jiawei Zhang¹, Bo Chen², and Yinyu Ye³

¹ Department of Management Science and Engineering, School of Engineering,
Stanford University, Stanford, CA 94305, USA
jiazhang@stanford.edu

² Warwick Business School, The University of Warwick, Coventry, England, UK
B.Chen@warwick.ac.uk

³ Management Science and Engineering and, by courtesy, Electrical Engineering,
Stanford University, Stanford, CA 94305, USA
yinyu-ye@stanford.edu

Abstract. We present a multi-exchange local search algorithm for approximating the capacitated facility location problem (CFLP), where a new local improvement operation is introduced that possibly exchanges multiple facilities simultaneously. We give a tight analysis for our algorithm and show that the performance guarantee of the algorithm is between $3 + 2\sqrt{2} - \epsilon$ and $3 + 2\sqrt{2} + \epsilon$ for any given constant $\epsilon > 0$. Previously known best approximation ratio for the CFLP is 7.88, due to Mahdian and Pál (2003), based on the operations proposed by Pál, Tardos and Wexler (2001). Our upper bound $3 + 2\sqrt{2} + \epsilon$ also matches the best known ratio, obtained by Chudak and Williamson (1999), for the CFLP with uniform capacities. In order to obtain the tight bound of our new algorithm, we make interesting use of the notion of exchange graph of Pál et al. and techniques from the area of parallel machine scheduling.

Key words: capacitated facility location, approximation algorithm, local search algorithm.

1 Introduction

In the capacitated facility location problem (CFLP), we are given a set of clients \mathcal{D} and a set of facilities \mathcal{F} . Each client $j \in \mathcal{D}$ has a demand d_j that must be served by one or more open facilities. Each facility $i \in \mathcal{F}$ is specified by a cost f_i , which is incurred when facility i is open, and by a capacity u_i , which is the maximum demand that facility i can serve. The cost for serving one unit demand of client j from facility i is c_{ij} . Here, we wish to open a subset of facilities such that the demands of all the clients are met by the open facilities and the total

* A full version of this paper including all the proofs is available at: www.stanford.edu/~jiazhang/ZhChYe.ps.

cost of facility opening and client service is minimized. We assume that unit service costs are non-negative, symmetric, and satisfy the triangle inequality, i.e., for each $i, j, k \in \mathcal{D} \cup \mathcal{F}$, $c_{ij} \geq 0$, $c_{ij} = c_{ji}$ and $c_{ij} \leq c_{ik} + c_{kj}$.

The CFLP is a well-known NP-hard problem in combinatorial optimization with a large variety of applications, such as location and distribution planning, telecommunication network design, etc. Numerous heuristics and exact algorithms have been proposed for solving CFLP. In this paper, we are concerned with approximation algorithms for the CFLP. Given a minimization problem, an algorithm is said to be a (polynomial) ρ -approximation algorithm, if for any instance of the problem, the algorithm runs in polynomial time and outputs a solution that has a cost at most $\rho \geq 1$ times the minimal cost, where ρ is called the performance guarantee or the *approximation ratio* of the algorithm.

Since the work of Shmoys, Tardos and Aardal (1997), designing approximation algorithms for facility location problems has received considerable attention during the past few years. In particular, almost all known approximation techniques, including linear programming relaxation, have been applied to the uncapacitated facility location problem (UFLP), which is a special case of the CFLP with all $u_i = \infty$. The best currently known approximation ratio for the UFLP is 1.517 due to Mahdian, Ye and Zhang (2002 and 2003). Furthermore, a result of Guha and Khuller (1999), combined with an observation of Sviridenko (1998) implies that no polynomial time algorithm for the UFLP can have a performance guarantee better than 1.463 unless $P = NP$.

On the other hand, because the (natural) linear programming relaxation for the CFLP is known to have an unbounded integrality gap, all known approximation algorithms for the CFLP with bounded performance guarantee are based on local search techniques. Korupolu, Plaxton and Rajaraman (1998) were the first to show that local search algorithm proposed by Kuehn and Hamburger (1963) has an approximation ratio of 8 for the case where all the capacities are uniform, i.e., $u_i = u$ for all $i \in \mathcal{F}$. The ratio has been improved to 5.83 by Chudak and Williamson (1998) using a simplified analysis. When the capacities are not uniform, Pál et al. (2001) have recently presented a local search algorithm with performance guarantee between 4 and 8.53. Their algorithm consists of the following three types of local improvement operations: open one facility; open one facility and close one or more facilities; close one facility and open one or more facilities. Very recently, Mahdian and Pál (2003) reduced the upper bound to 7.88 by using operations that combine those in Pál et al. (2001).

The main contribution of this paper is the introduction of a new type of operations, called the multi-exchange operations, which possibly exchanges multiple facilities simultaneously, i.e., close many facilities and open many facilities. In general, the multi-exchange operation cannot be computed in polynomial time since it is NP-hard to determine if such an operation exists so that a solution can be improved. However, we will restrict our attention to a subset of these operations such as the aforementioned determination can be computed efficiently. We will show that the restricted multi-exchange operation generalizes those in Pál et al. (2001).

This new type of operations enables us to approximate the CFLP with a factor of $3 + 2\sqrt{2} + \epsilon$ for any given constant $\epsilon > 0$. The approximation ratio of our algorithm matches what has been proved by Chudak and Williamson (1999) for the uniform capacity case. We also show that our analysis is almost tight. In particular, we construct an instance of the CFLP such that the ratio of the cost resulted from the improved local search algorithm to the optimal cost is $3 + 2\sqrt{2} - \epsilon$ for any given constant $\epsilon > 0$.

The performance guarantee of our algorithm follows from a set of inequalities that is implied by the local optimality of the solution, i.e., none of the legitimate operations can improve the solution. The multi-exchange operations are meant to give stronger inequalities than those implied by simpler operations. The major task of the proof is to identify those inequalities, which involves partitioning the facilities into groups. Here, we make uses of proof techniques from the area of parallel machines scheduling. Our proof also relies on the notion of exchange graph of Pál et al. (2001).

The remainder of this paper is organized as follows. In Section 2, we present our new local search operation together with some preliminary analyses. The performance guarantee of the algorithm is proved in Section 3. We present the lower bound ratio example in Section 4 and then make some remarks in the final section.

2 Local Search Algorithm

Given a subset $S \subseteq \mathcal{F}$, denote by $C(S)$ the optimal value of the following linear program $\text{LP}(S)$:

$$\begin{aligned} \text{LP}(S): \quad & \text{Minimize} \sum_{i \in S} f_i + \sum_{i \in S} \sum_{j \in \mathcal{D}} c_{ij} x_{ij} \\ & \text{subject to} \sum_{i \in S} x_{ij} = d_j \quad \forall j \in \mathcal{D} \\ & \quad \sum_{j \in \mathcal{D}} x_{ij} \leq u_i \quad \forall i \in S \\ & \quad x_{ij} \geq 0 \quad \forall i \in S, j \in \mathcal{D} \end{aligned}$$

where x_{ij} denotes the amount of demand of client j served by facility i . The above linear program $\text{LP}(S)$ is feasible if and only if $\sum_{i \in S} u_i \geq \sum_{j \in \mathcal{D}} d_j$. If it is infeasible, we denote $C(S) = +\infty$. If it is feasible, then an optimal solution $x_S = (x_{ij})$ and the optimal value $C(S)$ can be found in polynomial time.

Therefore, the CFLP is reduced to finding a subset $S \subseteq \mathcal{F}$, which we simply call a *solution* with service allocation x_S , such that $C(S)$ is minimized. Let $C_f(S) = \sum_{i \in S} f_i$ and $C_s(S) = C(S) - C_f(S)$ denote the respective facility cost and service cost associated with solution S .

Given the current solution S , we define for each facility $i \in \mathcal{F}$, $\bar{f}_i = f_i$ if $i \notin S$, and $\bar{f}_i = 0$ otherwise. We also let \bar{u}_i be the unused capacity of facility i

in the current solution S , i.e., $\bar{u}_i = u_i - \sum_{j \in \mathcal{D}} x_{ij}$ where x_{ij} is the amount of demand of client j served by facility i in solution S . For $i \notin S$, we define $\bar{u}_i = u_i$. The algorithm proposed by Pál et al. (2001) performs the following three types of local improvement operations:

- **add(s)**: Open a facility $s \in \mathcal{F} \setminus S$ and reassign all the demands to $S \cup \{s\}$ by computing $x_{S \cup \{s\}}$ and $C(S \cup \{s\})$ using linear program $\text{LP}(S \cup \{s\})$. The cost of this operation is defined as $C(S \cup \{s\}) - C(S)$, which can be computed in polynomial time for each $s \in \mathcal{F} \setminus S$.
- **open(s, T)**: Open a facility $s \in \mathcal{F}$, close a set of facilities $T \subseteq S \setminus \{s\}$ and reassign all the demands served by T to facility s . Demands served by facilities in $S \setminus T$ will not be affected. Notice that it is possible that $s \in S$, and in this case facility s only has capacity \bar{u}_s that can be used to serve the demands originally served by T . Let the *cost* of reassigning one unit of demand of any client j from facility $t \in T$ to facility s be c_{ts} , which is an upper bound on the change of service cost $c_{js} - c_{jt}$. The cost of operation **open(s, T)** is defined as the sum of $(\bar{f}_s - \sum_{i \in T} f_i)$ and the total cost of reassigning demands served by facilities in T to s . Pál et al. (2001) have shown that, for each $s \in \mathcal{F}$, if there exists a set T such that the cost of **open(s, T)** is $\bar{f}_s - \alpha$ for some $\alpha \geq 0$, then one can find in polynomial time a set T' such that the cost of **open(s, T')** is at most $\bar{f}_s - (1 - \epsilon)\alpha$ for any given $\epsilon > 0$.
- **close(s, T)**: Close a facility $s \in S$, open a set of facilities $T \subseteq \mathcal{F} \setminus \{s\}$ ($T \cap S$ may not be empty), and reassign all the demands served by s to facilities in T . Demands served by facilities in $S \setminus \{s\}$ will not be affected. Again let the cost of reassigning one unit of demand of any client j from facility s to facility $t \in T$ be c_{st} , which is an upper bound of $c_{jt} - c_{js}$. The cost of operation **close(s, T)** is defined as the sum of $(\sum_{i \in T} f_i - f_s)$ and the total cost of reassigning demands served by s to facilities in T . Similarly, for each $s \in S$, if there exists a set T such that the cost of **close(s, T)** is $\alpha - f_s$ for some $\alpha \geq 0$, then one can find in polynomial time a set T' such that the cost of **close(s, T')** is at most $(1 + \epsilon)\alpha - f_s$ for any given $\epsilon > 0$.

In our algorithm, we introduce an additional type of powerful restricted multi-exchange operations, which is defined below and consists of three parts.

- **multi(r, R, t, T)**: (i) Close a set of facilities $\{r\} \cup R \subseteq S$ where $r \notin R$ and R may be empty; (ii) Open a set of facilities $\{t\} \cup T \subseteq \mathcal{F} \setminus (\{r\} \cup R)$ where $t \notin T$, T may be empty, and $T \cap S$ may not be empty; (iii) Reassign the demands served by facilities in R to facility t and the demands served by facility r to $\{t\} \cup T$, and for each facility $i \in \{t\} \cup T$, it only has capacity \bar{u}_i to serve demands originally served by $\{r\} \cup R$. Recall that we let c_{st} be the cost of reassigning one unit of demand from facility $s \in \{r\} \cup R$ to facility t and c_{rt} be the cost of reassigning one unit of demand from facility r to facility $t \in T$. Then, the cost of operation **multi(r, R, t, T)** is defined as

the sum of $(\bar{f}_t + \sum_{i \in T} \bar{f}_i - f_r - \sum_{i \in R} f_i)$ and the total cost of reassigning (according to the given rules) demands served by facilities in $\{r\} \cup R$ to facilities in $\{t\} \cup T$, the latter of which will be called the service cost of operation $\text{multi}(r, R, t, T)$.

It is clear that $\text{multi}(r, R, t, T)$ generalizes $\text{open}(t, R \cup \{r\})$ by letting $T = \emptyset$. Operation $\text{multi}(r, R, t, T)$ also generalizes $\text{close}(r, T \cup \{t\})$ by letting $R = \emptyset$. Thus, the service costs of the operations $\text{open}(t, R)$ and $\text{close}(r, T)$ are also well defined. However, we will still use both notions $\text{open}(t, R)$ and $\text{close}(r, T)$ to distinguish them from the more general and powerful operation $\text{multi}(r, R, t, T)$. In the following lemma, we show that the restricted multi-exchange operation can also be implemented efficiently and effectively.

Lemma 1 *For any $r \in S$ and $t \in \mathcal{F} \setminus \{r\}$, if there exist a set $R \subseteq S \setminus \{r\}$ and a set $T \subseteq \mathcal{F} \setminus (\{r\} \cup R)$ such that the cost of $\text{multi}(r, R, t, T)$ is $A - B$ for some $A \geq 0$ and $B = f_r + \sum_{s \in R} f_s$, then one can find in polynomial time sets R' and T' such that the cost of $\text{multi}(r, R', t, T')$ is at most $(1 + \epsilon)A - (1 - \epsilon)B$ for any given $\epsilon > 0$.*

In order to prove this lemma, we need the following technical result.

Proposition 1 *Let (x^*, z^*) be an optimal basic feasible solution for the following linear program:*

$$\text{maximize} \quad f(x, z) \quad (1)$$

$$\text{subject to} \quad \sum_{i \in I} d_i z_i \leq u \quad (2)$$

$$a \sum_{i \in I} d_i z_i + b \leq \sum_{i \in I} c_i x_i \leq a' \sum_{i \in I} d_i z_i + b' \quad (3)$$

$$z_i + x_i \leq 1 \quad \forall i \in I \quad (4)$$

$$z_i, x_i \geq 0, \quad (5)$$

where $f(x, z)$ is a linear function of (x, z) . Let

$$J_1 = \{i \in I | 0 < x_i^* < 1\} \quad \text{and} \quad J_2 = \{i \in I | 0 < z_i^* < 1\}.$$

Then $|J_1|, |J_2| \leq 2$. In other words, the optimal solution (x^*, z^*) has at most 2 fractional values among its x and z components, respectively.

Proof. An inequality is said to be tight if it holds as an equality at (x^*, z^*) . It is evident that at most two inequalities in constraints (2) and (3) are tight. We assume that there are $m \leq |I|$ inequalities in (4) that are tight, which implies that at most $m + 2$ of the variables x_i^* and z_i^* ($i = 1, 2, \dots, |I|$) are positive since (x^*, z^*) is a basic feasible solution.

Observe that each of the m tight inequalities in (4) contains at least one positive variables. Therefore, if we let m' be the number of tight inequalities in (4) that contain more than one positive variable each, then $m' \leq 2$ and $m - m'$ tight constraints in (4) are in the form of either $x_i^* = 1$ or $z_i^* = 1$. In the case

of $m' = 2$, we conclude that $J_1 = J_2$ and they contain two elements each. It is easy to see that $|J_1|, |J_2| \leq 2$ in each of the cases of $m' = 0$ and $m' = 1$. ■

Remarks. In the full version of this paper, we show that the problem of finding the best operation $\text{multi}(r, R, t, T)$ can be formulated as a $\{0, 1\}$ -integer program, which is the same as (1)-(5) except that we require $x_i, z_i \in \{0, 1\}$. One can see that, if we ignore constraints in (4), then by guessing an appropriate value of $\sum_{i \in I} d_i z_i$, the restricted multi-exchange operation can be decoupled into two $\{0, 1\}$ -integer programs, and each of them contains either only x variables or only z variables. These two integer programs correspond to operations **close** and **open**, respectively. A basic feasible solution for the linear programming relaxation of any of these two integer programs would have at most one fractional value, which essentially implies that one can approximate the restricted multi-exchange operation efficiently and effectively by a simple combinatorial procedure. Now, without ignoring constraints in (4) in the linear program, Proposition 1 guarantees that the restricted multi-exchange operation can still be approximated efficiently and effectively.

Starting with any feasible solution S , our local search algorithm repeatedly performs the legitimate four types of operations defined above until none of them is *admissible*, where an operation is said to be admissible if its cost is less than $-C(S)/p(\|\mathcal{D}\|, \epsilon)$ with $p(\|\mathcal{D}\|, \epsilon) = 3\|\mathcal{D}\|/\epsilon$. Then it is clear that the algorithm will terminate after at most $O(p(\|\mathcal{D}\|, \epsilon) \log(C(S)/C(S^*)))$ operations. A solution is said to be a *local optimum* if none of the four legitimate operations will have a negative cost. We will call a solution *an approximately locally optimal solution* if none of the four legitimate operations is admissible. We summarize results in this section in the following theorem.

Theorem 1 *The multi-exchange local search algorithm can identify and implement any admissible operation in polynomial time for any given $\epsilon > 0$. The algorithm terminates in polynomial time and outputs an approximately locally optimal solution.*

3 Analysis of the Algorithm

Now we are ready to analyze our algorithm. Let $S, S^* \subseteq \mathcal{F}$ be any local and global optimal solutions, respectively. As is done in Korupolu et al (1998), Chudak and Williamson (1999), Pál et al. (2001), and Mahdian and Pál (2003), we will bound the facility cost $C_f(S)$ and service cost $C_s(S)$ separately. The analysis of our algorithm mainly follows that of Pál et al. (2001). In particular, we use their notion of ‘exchange graph’. However, we take advantage of the availability of our newly introduced type of multi-exchange operations and use some techniques from parallel machine scheduling. Our analysis is based on a new classification and partition of the nodes of the exchange graph, and a new construction of a series of operations to cover these nodes.

As stated in Mahdian and Pál (2003), any local search algorithm with **add** operation in its repertoire will guarantee a low service cost when it settles down

to a local optimum, which is summarized in the following lemma, various versions of which have been proved in the aforementioned papers. We omit its proof here.

Lemma 2 *The service cost $C_s(S)$ is bounded above by the optimal total cost $C(S^*)$.*

Let x_{ij} denote the amount of demand of client j served by facility i in solution S , and denote $x(i, \cdot) = \sum_{j \in \mathcal{D}} x_{ij}$. Using a simple argument of network flow, Pál, Tardos and Wexler (2001) have proved the following.

Lemma 3 *The optimal value of the following linear program is at most $C_s(S^*) + C_s(S)$:*

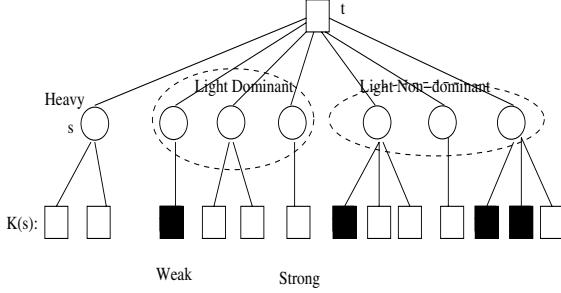
$$\begin{aligned} & \text{minimize} \quad \sum_{s \in S \setminus S^*} \sum_{t \in S^*} c_{st} y(s, t) \\ & \text{subject to} \quad \sum_{t \in S^*} y(s, t) = x(s, \cdot) \quad \forall s \in S \setminus S^* \\ & \quad \sum_{s \in S \setminus S^*} y(s, t) \leq u_t - x(t, \cdot) \quad \forall t \in S^* \\ & \quad y(s, t) \geq 0 \quad \forall s \in S \setminus S^*, t \in S^*. \end{aligned} \tag{6}$$

For notational clarity, we denote $c(s, t) = c_{st}$. Let $y(s, t)$ be the optimal solution of (6). Consider bipartite graph $G = (S \setminus S^*, S^*, E)$, where $S \setminus S^*$ and S^* are the two sets of nodes in the bipartite graph and $E = \{(s, t) : s \in S \setminus S^*, t \in S^* \text{ and } y(s, t) > 0\}$ is the set of edges. Due to the optimality of $y(s, t)$, we can assume without loss of generality that G is a forest.

Consider any tree \mathcal{T}_r of G , which is rooted at node $r \in S^*$. It is evident that the tree has alternate layers of nodes in S^* and in $S \setminus S^*$. For each node t in this tree, let $K(t)$ be the set of children of t . For any $t \in S^*$ and any $s \in S \setminus S^*$, denote $y(\cdot, t) = \sum_{s \in S \setminus S^*} y(s, t)$ and $y(s, \cdot) = \sum_{t \in S^*} y(s, t)$. A facility node $t \in S^*$ is said to be *weak* if $\sum_{s \in K(t)} y(s, t) > \frac{1}{2}y(\cdot, t)$, and *strong* otherwise.

Let us concentrate on any given sub-tree of depth at most 2 rooted at $t \in S^*$, which consists of t , $K(t)$ and $G(t) = \bigcup_{s \in K(t)} K(s)$. A node $s \in K(t)$ is said to be *heavy* if $y(s, t) > \frac{1}{2}y(\cdot, t)$, and *light* otherwise. A light node $s \in K(t)$ is said to be *dominant* if $y(s, t) \geq \frac{1}{2}y(s, \cdot)$, and *non-dominant* otherwise. The set $K(t)$ is partitioned into three subsets: the set $H(t)$ of heavy nodes, the set $Dom(t)$ of light and dominant nodes and the set $NDom(t)$ of light and non-dominant nodes (see Figure 1 for an illustration). It is clear that if $t \in S^*$ is strong, then $H(t)$ must be empty.

We are to present a list of legitimate operations such that (i) each facility in $S \setminus S^*$ is closed exactly once, (ii) every facility in S^* is opened a small number of times, and (iii) the total reassignment cost is not too high. Depending on their types, the nodes in $S \setminus S^*$ and S^* will be involved in different operations.

**Fig. 1.** Facility classification

Let us first deal with the nodes in $NDom(t)$. For each node $s \in NDom(t)$, let $W(s) = \{t' \in K(s) : t' \text{ is weak}\}$ and

$$Rem(s) = \max \left\{ y(s, t) - \sum_{t' \in W(s)} y(s, t'), \quad 0 \right\}.$$

Re-index the nodes in $NDom(t)$ as s_1, s_2, \dots, s_k for some $k \geq 0$ ($NDom(t) = \emptyset$ if $k = 0$), such that $Rem(s_i) \leq Rem(s_{i+1})$ for $i = 1, \dots, k-1$. The operations given below will close s_i exactly once for $1 \leq i < k$. We will consider node s_k together with nodes in $Dom(t)$.

We perform operation $\text{close}(s_i, K(s_i) \cup (K(s_{i+1}) \setminus W(s_{i+1})))$ for any $i = 1, \dots, k-1$. The feasibility and costs of these operations are established in the following lemma.

Lemma 4 For $i = 1, 2, \dots, k-1$, the following hold:

(i) Operation $\text{close}(s_i, K(s_i) \cup (K(s_{i+1}) \setminus W(s_{i+1})))$ is feasible, i.e., the total capacity of facilities opened by the operation is at least the total capacity of those closed by the operation:

$$y(s_i, \cdot) \leq \sum_{t' \in K(s_i) \cup (K(s_{i+1}) \setminus W(s_{i+1}))} y(\cdot, t').$$

(ii) The service cost of operation $\text{close}(s_i, K(s_i) \cup (K(s_{i+1}) \setminus W(s_{i+1})))$ is at most

$$\begin{aligned} & c(s_i, t)y(s_i, t) + \sum_{t' \in W(s_i)} 2c(s_i, t')y(s_i, t') + \sum_{t' \in K(s_i) \setminus W(s_i)} c(s_i, t')y(s_i, t') \\ & + c(s_{i+1}, t)y(s_{i+1}, t) + \sum_{t' \in K(s_{i+1}) \setminus W(s_{i+1})} c(s_{i+1}, t')y(s_{i+1}, t'). \end{aligned}$$

Now we move to considering nodes in $H(t)$, $Dom(t)$ and the node s_k . To do this, we need to distinguish several cases on t and $H(t)$.

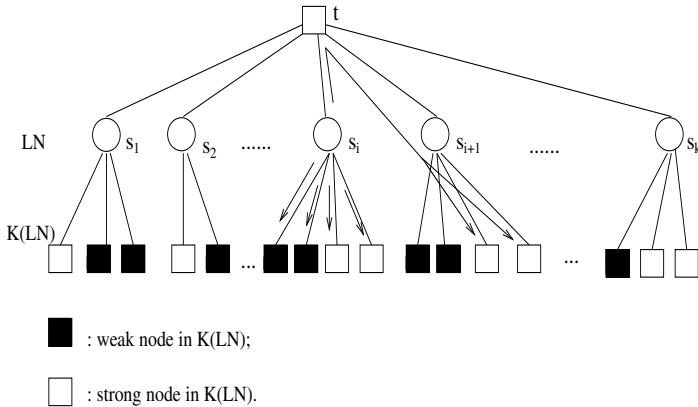


Fig. 2. Illustration of operation $\text{close}(s_i, K(s_i) \cup (K(s_{i+1}) \setminus W(s_{i+1})))$

Case 1. Node t is strong (see Figure 3 for illustration). This is an easy case. As we observed before, we have $H(t) = \emptyset$. Perform operation $\text{multi}(s_k, Dom(t), t, K(s_k))$. A feasible reassignment to t of demands served by s_k and facilities in $Dom(t)$ and $K(s_k)$ is as follows: for each $s \in Dom(t)$, reassign $y(s, \cdot)$ amount of demand to t ; for s_k , reassign $y(s_k, t)$ amount of demand to t and $y(s_k, t')$ amount of demand to $t' \in K(s_k)$. To show that the operation is feasible, it suffices to show that $\sum_{s \in Dom(t)} y(s, \cdot) + y(s_k, t) \leq y(\cdot, t)$ and $y(s_k, t') \leq y(\cdot, t')$ for any $t' \in K(s_k)$. The second is trivial and the first holds since $s \in Dom(t)$ and t is strong, and hence

$$\sum_{s \in Dom(t)} y(s, \cdot) + y(s_k, t) \leq 2 \sum_{s \in Dom(t)} y(s, t) + y(s_k, t) \leq y(\cdot, t).$$

This leads to the following lemma.

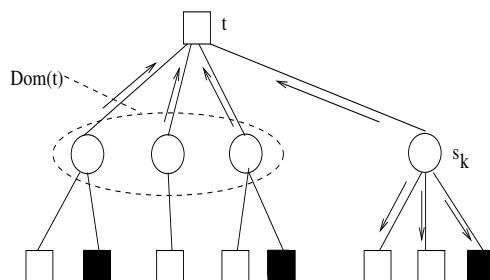


Fig. 3. Illustration of Case 1

Lemma 5 If t is strong, then operation $\text{multi}(s_k, \text{Dom}(t), t, K(s_k))$ is feasible, and the service cost of this operation is at most

$$\sum_{s \in \text{Dom}(t)} 2c(s, t)y(s, t) + c(s_k, t)y(s_k, t) + \sum_{t' \in K(s_k)} c(s_k, t')y(s_k, t').$$

Case 2. Node t is weak and $H(t) \neq \emptyset$ (see Figure 4 for illustration). Since there can be no more than one heavy node, let $H(t) = \{s_0\}$. We perform operations $\text{close}(s_0, \{t\} \cup K(s_0))$ and $\text{multi}(s_k, \text{Dom}(t), t, K(s_k))$. For operation $\text{multi}(s_k, \text{Dom}(t), t, K(s_k))$, the reassignment of demands is done exactly as in Case 1. For operations $\text{close}(s_0, \{t\} \cup K(s_0))$, we reassign $y(s_0, t)$ amount of demand to t and $y(s_0, t')$ amount of demand to $t' \in K(s_0)$, which is obviously feasible. Then we have

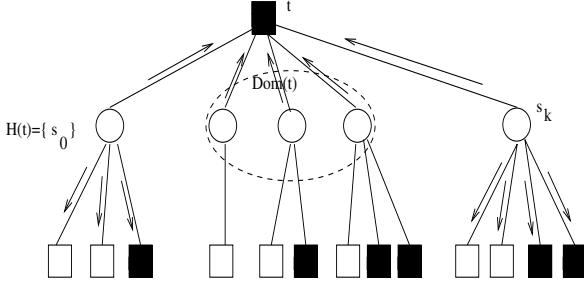


Fig. 4. Illustration of Case 2

Lemma 6 If node t is weak and $H(t) = \{s_0\}$, then

(i) Operation $\text{multi}(s_k, \text{Dom}(t), t, K(s_k))$ is feasible and the service cost of this operation is at most

$$\sum_{s \in \text{Dom}(t)} 2c(s, t)y(s, t) + c(s_k, t)y(s_k, t) + \sum_{t' \in K(s_k)} c(s_k, t')y(s_k, t').$$

(ii) Operation $\text{close}(s_0, \{t\} \cup K(s_0))$ is feasible and its service cost is at most

$$\sum_{t' \in K(s_0)} c(s_0, t')y(s_0, t') + c(s_0, t)y(s_0, t).$$

Case 3. Node t is weak and $H(t) = \emptyset$ (see Figure 5 for illustration). The analysis for this last case is more involved. In order to find feasible operations, we need the following lemma, where we make use of techniques from the area of parallel machine scheduling.

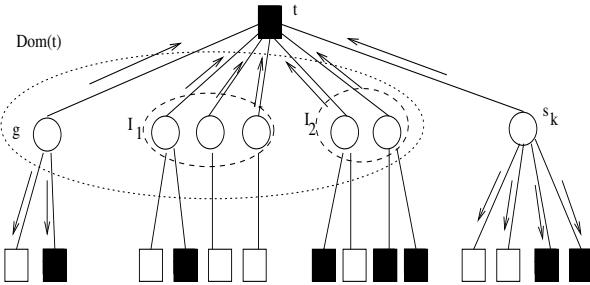


Fig. 5. Illustration of Case 3 ($g = \gamma_1$ in Lemma 7).

Lemma 7 *If $\text{Dom}(t) \neq \emptyset$, then there exists a node $\gamma_1 \in \text{Dom}(t)$ such that set $\text{Dom}(t) \setminus \{\gamma_1\}$ can be partitioned into two subsets D_1 and D_2 that satisfy the following:*

$$y(\gamma_i, t) + \sum_{s \in D_i} y(s, \cdot) \leq y(\cdot, t) \text{ for } i = 1, 2,$$

where $\gamma_2 = s_k$.

Lemma 8 *Let γ_i and D_i ($i = 1, 2$) be those defined in Lemma 7. If node t is weak and $H(t) = \emptyset$, then operations $\text{multi}(\gamma_i, D_i, t, K(\gamma_i))$ ($i = 1, 2$) are feasible, and their service costs are respectively at most*

$$\sum_{s \in D_i} 2c(s, t)y(s, t) + c(\gamma_i, t)y(\gamma_i, t) + \sum_{t' \in K(\gamma_i)} c(\gamma_i, t')y(\gamma_i, t')$$

for $i = 1, 2$.

Now we are ready to provide a global picture of the impact of all the operations and demand reassignment we have introduced, which will help bound the total facility cost of a local optimal solution.

Lemma 9 *Given any $r \in S^*$, the operations we defined on sub-tree \mathcal{T}_r and the corresponding reassignment of demands satisfy the following two conditions: (i) Each facility $s \in S \setminus S^*$ is closed exactly once; (ii) Every facility $t \in S^*$ is opened at most 3 times; (iii) The total reassignment cost of all the operations is at most $2 \sum_{s \in S \setminus S^*} \sum_{t \in S^*} c(s, t)y(s, t)$.*

Proof. The bounds on the total reassignment cost follow from Lemmas 4,5,6 and 8. We show that every facility $t \in S^*$ is opened at most 3 times. Note that when we take all sub-trees of \mathcal{T}_r of depth 2 into account, any $t \in S^*$ can be the root of a subtree at most once and a leaf of a sub-tree at most once. If $t \in S^*$ is strong, it will be opened once as a root and twice as a leaf. If t is weak, it will be opened at most once as a leaf and at most twice as a root. ■

Theorem 2 $C(S) \leq 6C_f(S^*) + 5C_s(S^*)$.

Proof. Since S is locally optimal, none of the above defined operations has a negative cost. Therefore, the total cost will also be non-negative. It follows from Lemma 9 that

$$3C_f(S^* \setminus S) + 2 \sum_{s \in S \setminus S^*} \sum_{t \in S^*} c(s, t)y(s, t) - C_f(S \setminus S^*) \geq 0.$$

The theorem follows from Lemmas 2 and 3. ■

Scaling the service costs c_{ij} by a factor $\delta = (1 + \sqrt{2})/2$ and with Lemma 2 and Theorem 2, we can show that $C(S) \leq (3 + 2\sqrt{2})C(S^*)$ which, together with Theorem 1, leads to the following corollary.

Corollary 1 *For any small constant $\epsilon > 0$, the multi-exchange local search algorithm runs in polynomial time with approximation guarantee of $3 + 2\sqrt{2} + \epsilon \leq 5.83$.*

4 A Tight Example

Figure 6 illustrates an example that gives a lower bound of our algorithm. In this example, we have a total of $4n$ clients that are represented by triangles, of which $2n$ have demands $n - 1$ each and the other $2n$ have demands 1 each. There are a total of $4n + 1$ facilities. In the current solution, the $2n$ open facilities are represented by squares, each of which has facility cost 4 and capacity n . The other $2n + 1$ facilities are represented by circles, of which the one at the top of the figure has facility cost 4 and capacity $2n$, and each of the other $2n$ facilities at the bottom of the figure has facility cost 0 and capacity $n - 1$. The numbers by the edges represent the unit service costs scaled by the factor of $\delta = (1 + \sqrt{2})/2$. Other unit service costs are given by the shortest distances induced by the given edges. We will call a facility *circle-facility* or *square-facility* depending on how it is represented in the figure.

We show that the current solution is a local optimum, i.e., none of the four operations will have a negative cost. First of all, for **add** operation, it is easy to see that open any of the circle-facilities would not reduce the total service cost. Secondly, notice that the connection cost between a square-facility and a circle-facility is 0, 2, or 4. Therefore, closing any of the square-facilities would require opening the top circle-facility, and hence any **close** operation will not have a negative cost. Thirdly, we cannot open one bottom circle-facility while closing any of the square-facilities due to the capacity constraint. Therefore, in order to open one facility and close one or many facilities, we have to open the top circle-facility. In this case, we could possibly close any two of the square-facilities. However, routing one unit of demand from any square-facility to the top circle-facility will have a cost of 2. Therefore, any **open** operation will not be profitable. Finally, we show that the restricted multi-exchange operation **multi**(r, R, t, T) is

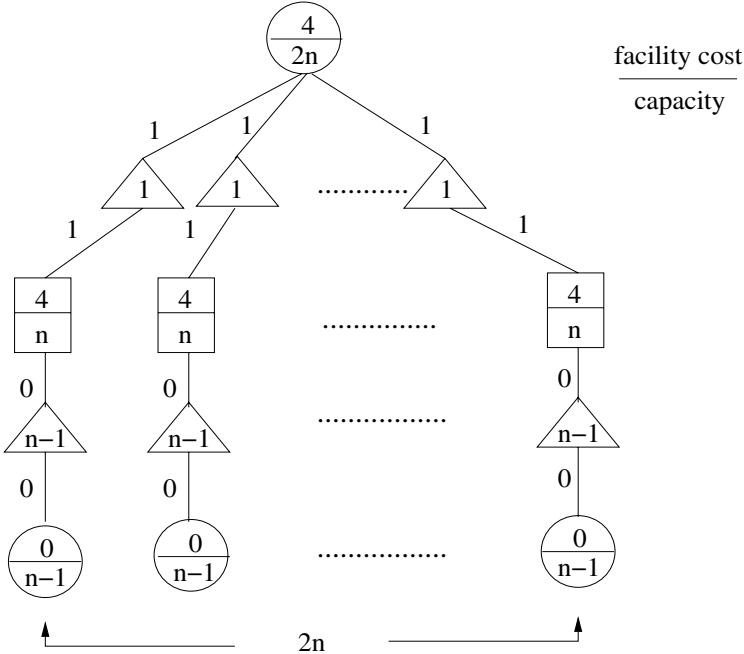


Fig. 6. A tight example with performance ratio $3 + 2\sqrt{2} - \epsilon$

not profitable either. We have already shown that this is the case $R = \emptyset$ or $T = \emptyset$. Therefore, we assume that $R \neq \emptyset$ and $T \neq \emptyset$. By symmetry, we can assume r is any of the square-facilities. Since the demands served by R will be routed to t , capacity constraint forces t to be the top circle-facility. By comparing the capacities of the top circle-facility and the square-facilities, we know that $|R| \leq 2$. Now our final claim follows from the fact that routing one unit of demand from the square-facility to the circle-facility will cost 2. Therefore, the current solution is indeed a local optimum.

The current solution has a total facility cost of $8n$ and a total service cost of $2n/\delta = 4n/(1 + \sqrt{2})$ in terms of the original (pre-scaling) data. However, if we open all the circle-facilities only, then the total facility cost is 4 and the total service cost is $4n/(1 + \sqrt{2})$. The total costs of these two solutions are $4n(3 + 2\sqrt{2})/(1 + \sqrt{2})$ and $4n/(1 + \sqrt{2}) + 4$, respectively, and their ratio approaches $3 + 2\sqrt{2}$ when n goes to ∞ , which provides the desired lower bound of our algorithm.

5 Final Remarks

A new local search algorithm for the CFLP has been proposed in this paper. The performance guarantee of the algorithm is shown to be between $3 + 2\sqrt{2} - \epsilon$ and $3 + 2\sqrt{2} + \epsilon$ for any constant $\epsilon > 0$. This is the currently best known approximation

ratio for the CFLP. Moreover, it is the first time that a tight bound of a local search algorithm for the CFLP is developed. Our result is based not only on the introduction of a new type of multi-exchange local improvement, but also on the exploitation of a technique from the area of parallel machine scheduling.

Several significant questions remain open. It is known that the UFLP can not be approximated better than 1.463 in polynomial time and this lower bound naturally also applies to the more general CFLP. But can better (larger) lower bound be established for the CFLP? On the other hand, it would also be challenging to improve the upper bound $3 + 2\sqrt{2} + \epsilon$. In order to obtain a better approximation ratio, it seems that new local improvement procedures are needed. Furthermore, it is known that the natural linear programming relaxation has an unbounded integrality gap. It would be interesting to develop new LP relaxations for the CFLP such that the known techniques for the UFLP such as randomized rounding, primal-dual, and dual-fitting can be applied to the CFLP.

Many other variants of the UFLP have been studied in the literature, e.g., k -median (Arya et al 2001), multi-level facility location (Aardal, Chuak, and Shmoys 1999, Ageev, Ye, and Zhang 2003, Zhang 2004), fault-tolerant facility location (Guha, Meyerson, and Munagala 2001, Swamy and Shmoys 2003), facility location with outliers (Charikar et al 2001). No constant approximation ratio is known for any of these problems with capacity constraints (Khuller and Sussman 2000 developed a 6-approximation algorithm for the capacitated k -center problem). Since the presence of capacity constraints is natural in practice, it would be important to study all of these problems with capacity constraints.

Acknowledgements The full version of this paper has been submitted to *Mathematics of Operations Research*. The authors would like to thank the area editor and two anonymous referees for many useful comments on a previous version of the paper. In particular, one of the referees pointed out a technical error in an earlier version of our proof of Lemma 1, in which we implicitly overlooked the constraints in (4). And this error has now been fixed.

References

1. K. Aardal, F. A. Chudak and D. B. Shmoys, “A 3-Approximation Algorithm for the k -Level Uncapacitated Facility Location Problem,” *Information Processing Letters* 72(5-6): 161-167, 1999.
2. A. Ageev, Y. Ye and J. Zhang, “Improved Combinatorial Approximation Algorithms for the k -Level Facility Location Problem,” in *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (30th ICALP)*, LNCS 2719, 145-156, 2003.
3. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala and V. Pandit, “Local search heuristic for k -median and facility location problems,” in *Proceedings of the ACM Symposium on Theory of Computing*, 21-29, 2001.
4. M. Charikar and S. Guha, “Improved combinatorial algorithms for facility location and k -median problems,” in *Proceedings of the 40th IEEE Foundations of Computer Science*, 378-388, 1999.

5. M. Charikar, S. Khuller, D. M. Mount and G. Narasimhan, "Algorithms for facility location problems with outliers," in *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, 642-651, 2001.
6. G. Cornuéjols, G.L. Nemhauser and L.A. Wolsey, "The uncapacitated facility location problem," in: P. Mirchandani, R. Francis (Eds.), *Discrete Location Theory*, Wiley, New York, 119-171, 1990.
7. F.A. Chudak and D.B Shmoys, "Improved approximation algorithms for the uncapacitated facility location problem," *SIAM J. on Computing*, to appear.
8. F. A. Chudak and D. Williamson, "Improved approximation algorithms for capacitated facility location problems," in *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization (IPCO'99)*, 99-113, 1999.
9. S. Guha and S. Khuller, "Greedy strikes back: improved facility location algorithms," *Journal of Algorithms*, 228-248, 1999.
10. S. Guha, A. Meyerson and K. Munagala, "Improved algorithms for fault tolerant facility location," in *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, 636-641, 2001.
11. S. Khuller and Y. J. Sussmann, "The Capacitated K-Center Problem," *SIAM Journal on Discrete Mathematics* 13(3): 403-418, 2000.
12. M. R. Korupolu, C. G. Plaxton and R. Rajaraman, "Analysis of a Local Search Heuristic for Facility Location Problems," *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1-10, 1998.
13. A. A. Kuehn and M. J. Hamburger, "A heuristic program for locating warehouses," *Management Science*, 9:643-666, 1963.
14. K. Jain, M. Mahdian, and A. Saberi, "A new greedy approach for facility location problems," in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, 731-740, 2002.
15. M. Mahdian and M. Pál, "Universal facility location," in *Proceedings of the 11th Annual European Symposium on Algorithms (ESA)*, 409-421, 2003.
16. M. Mahdian, Y. Ye and J. Zhang, "Improved approximation algorithms for metric facility location problems," in *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, LNCS 2462, 229-242, 2002.
17. M. Mahdian, Y. Ye and J. Zhang, "A 2-approximation algorithm for the soft-capacitated facility location problem," *Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, LNCS 2764, 129-140, 2003.
18. M. Pál, É. Tardos and T. Wexler, "Facility location with hard capacities," in *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 329-338, 2001.
19. D. B. Shmoys, "Approximation algorithms for facility location problems," *3rd International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, 27-33, 2000.
20. D.B. Shmoys, É. Tardos, and K.I. Aardal, "Approximation algorithms for facility location problems," in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 265-274, 1997.
21. M. Sviridenko, cited as personal communication in [8], July, 1998.
22. C. Swamy and D.B. Shmoys, "Fault-tolerant facility location," in *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 735-736, 2003.
23. J. Zhang, "Approximating the two-level facility location problem via a quasi-greedy approach," in *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages. 801-810, 2004.

Separable Concave Optimization Approximately Equals Piecewise Linear Optimization

Thomas L. Magnanti¹ and Dan Stratila²

¹ School of Engineering and Sloan School of Management,
Massachusetts Institute of Technology,
Room 1-206, 77 Massachusetts Avenue, Cambridge, MA 02139-4307
magnanti@mit.edu

² Operations Research Center, Massachusetts Institute of Technology,
Room E40-130, 77 Massachusetts Avenue, Cambridge, MA 02139-1309
dstrat@mit.edu

Abstract. We show how to approximate a separable concave minimization problem over a general closed ground set by a *single* piecewise linear minimization problem. The approximation is to arbitrary $1 + \epsilon$ precision in optimal cost. For polyhedral ground sets in \mathbb{R}_+^n and nondecreasing cost functions, the number of pieces is polynomial in the input size and proportional to $1/\log(1+\epsilon)$. For general polyhedra, the number of pieces is polynomial in the input size and the size of the zeroes of the concave objective components.

We illustrate our approach on the concave-cost uncapacitated multi-commodity flow problem. By formulating the resulting piecewise linear approximation problem as a fixed charge, mixed integer model and using a dual ascent solution procedure, we solve randomly generated instances to within five to twenty percent of guaranteed optimality. The problem instances contain up to 50 nodes, 500 edges, 1,225 commodities, and 1,250,000 flow variables.

1 Introduction

Minimizing a separable concave function over a feasible set arises often in practice due to economies of scale in transportation and production, fixed equipment costs, and other economic factors. A natural approach for solving these problems, or approximating any continuous function, is to replace the nonlinear function by a piecewise linear approximation, an idea known in the field of optimization since at least the 1950s. Today, many textbooks describe at least one way to model these approximations as mixed integer programs [1].

Clearly, to improve the quality of the approximation, we would increase the number of pieces. However, even when optimizing separable concave or convex functions, not much is known about the number of pieces required for a single approximation to attain a desired precision. Meyerson et al. [2], in the context of the concave cost single sink multicommodity flow, remark that a “tight” approximation could be computed. Munagala [3] states, in the same context, that

an approximation of arbitrary precision could be obtained with a polynomial number of pieces, without mentioning specific bounds, or any details on how to do so.

Most of the work for approximating separable concave or convex costs with linear pieces has used a scale-and-iterate approach: using an equally spaced grid, solve the approximate problem; then iteratively approximate the problem using an increasingly denser grid on a shrinking feasible region. The analysis uses the special structure of the problem and the properties of the algorithm to establish optimality or approximation results, and to bound the number of iterations and pieces on the grid. A classical example is the convex cost flow scaling algorithm (see, for example, [4]).

Hochbaum and Shanthikumar [5] have conducted perhaps the most general study of this approach. They consider separable convex costs over general polyhedra, and use a scale-and-iterate approach to obtain a $(1 + \epsilon)$ -approximate solution. Their algorithm is polynomial in the size of the input, and the absolute value $|\Delta|$ of the largest subdeterminant of the constraint matrix. However, they measure the approximation in terms of the solution vectors themselves, not the objective values. They suggest methods for achieving objective approximation, with a running time dependent on the behavior of the cost function, as well as the size of the input and $|\Delta|$.

1.1 Our Contribution

In contrast to the previous contributions, we show how to approximate separable concave functions to $1 + \epsilon$ precision, in optimal cost, using a *single* piecewise linear approximation for each component of the concave objective. The key idea that enables us to avoid iterations and scaling, and yet to obtain polynomial bounds, is to use pieces exponentially increasing in size. Because we approximate the objective value, we restrict the objective functions to be nonnegative.

In Section 2 we introduce our technique on general grounds sets. We need only $3 + 2 \left\lceil \log_{1+4\epsilon+4\epsilon^2} \frac{2u_i}{l_i} \right\rceil$ pieces for each concave component of the objective; in this expression, u_i denotes an upper bound on the value of corresponding variable, and l_i the smallest distance between a feasible value of that component and a zero of the concave function. As $\epsilon \rightarrow 0$, the number of pieces as a function of ϵ behaves as $\frac{1}{4\epsilon}$. The number of pieces is the same for any concave function, and depends only on the chosen value of ϵ and the bounds u_i and l_i . Our method requires just one function and derivative (or any supergradient) evaluation per piece.

The results for general ground sets provide a bridge between concave function minimization and piecewise linear minimization. Since our technique requires only a single piecewise linear approximation, we can directly apply any algorithm for optimizing piecewise linear objectives, without the need to develop special scaling-based algorithms.

In Section 3 we show that when we restrict the feasible set to polyhedra in \mathbb{R}_+^n and minimize nondecreasing concave functions, we need only a polynomial

number of pieces as a function of the input size; no other conditions are necessary. For general polyhedra and any nonnegative concave functions, we show that the number of required pieces is polynomial in the input size and the size of the zeroes of the cost functions. We assume the input size for the nonlinear concave minimization problems is the size of the polyhedral ground set when represented by linear inequalities.

In Section 3.1 we present a mixed integer programming formulation for the resulting piecewise linear optimization problem that often preserves the underlying structure (for example, network structure). The polynomial bounds on the size of the resulting problems imply that solution methods for piecewise linear optimization, and solution methods in integer programming and combinatorial optimization all become applicable for concave minimization problems. For practical problems, these advantages are amplified by the possibility of establishing significantly lower bounds on the number of pieces.

In Section 4 we illustrate our method on the practical and pervasive uncapacitated multicommodity flow problem with complete demand. We derive considerably smaller bounds on the number of required pieces than in the general case. Since our method preserves structure, the resulting fixed charge problems are network design problems. Using a dual ascent method [6], we are able to solve, to within five to twenty percent of optimality, large problems with up to 50 nodes, 500 edges, 1,225 commodities and 1,250,000 flow variables. These problems are, to the best of our knowledge, the largest general concave cost multicommodity flow problems with full demand anyone has solved, approximately or exactly.

In this paper, we first approximate a given concave optimization problem and then apply dual ascent to the resulting piecewise linear optimization problem. A potentially attractive alternative would be a dual ascent algorithm that introduces the pieces of the approximation only implicitly. For several common dual ascent procedures, the resulting “continuous” dual ascent algorithm terminates in polynomial time, and yields an approximation ratio independent of ϵ that is essentially the same as that of the classic dual ascent algorithm applied to the explicit piecewise linear problem obtained as $\epsilon \rightarrow 0$.

Finally, our technique is not limited even to the optimization framework of Section 2. It is potentially applicable for approximating problems in continuous dynamic programming, continuous optimal control, and other settings in which new solutions methods become available when switching from continuous to piecewise linear cost functions.

2 General Ground Sets

We examine the general concave minimization problem

$$\min \{f(x) : x \in X, x \geq 0\} \quad (1)$$

defined by closed ground set $X \subseteq \mathbb{R}^n$ and a separable concave cost function $f : \mathbb{R}_+^n \rightarrow \mathbb{R}$. That is, $f(x) = \sum_{i=1}^n f_i(x_i)$, with $x = (x_1, \dots, x_n)$. The ground

set need not be convex or connected. In this section, we impose the following assumption (later in this section, we state a more general result).

Assumption 1 (a) For $i \in \{1, \dots, n\}$ the functions f_i are nondecreasing in x_i and $f_i(0) \geq 0$. (b) The problem has an optimal solution x^* and bounds $0 < l_i \leq u_i$ so that for each $i \in \{1, \dots, n\}$ either $x_i^* = 0$ or $x_i^* \in [l_i, u_i]$.

To approximate problem (1) within a factor of $1 + \epsilon$, we approximate each function f_i with a piecewise linear function $g_i : \mathbb{R} \rightarrow \mathbb{R}$. Each function g_i consists of $1 + p_i := 1 + \lceil \log_{1+2\epsilon} \frac{u_i}{l_i} \rceil$ pieces defined by the coefficients

$$c_i^j := \frac{d}{dx} f_i(l_i(1+\epsilon)^j), \quad j \in \{0, \dots, p_i\}, \quad (2a)$$

$$d_i^j := f_i(l_i(1+\epsilon)^j) - l_i(1+\epsilon)^j c_i^j, \quad j \in \{0, \dots, p_i\}. \quad (2b)$$

The symbol $\frac{df_i(x'_i)}{dx_i}$ denotes the derivative of f_i at $x_i = x'_i$ if f_i is differentiable at x'_i , and an arbitrary supergradient of f_i at x'_i otherwise. Each coefficient pair defines a line with slope c_i^j and y-intercept d_i^j that is tangent to the graph of f_i at the point $l_i(1+\epsilon)^j$. For $x_i > 0$, the function g_i is defined by the lower envelope of these lines:

$$g_i(x_i) := \min\{d_i^j + c_i^j x_i : 1 \leq j \leq p_i\}. \quad (3)$$

We set $g_i(0) = f_i(0)$ and let $g(x) := \sum_{i=1}^n g_i(x_i)$. Substituting g for f , we obtain the piecewise linear concave minimization problem

$$\min\{g(x) : x \in X, x \geq 0\}. \quad (4)$$

Theorem 1 Let Z_1^* be the optimal cost of problem (1) and Z_4^* the optimal cost of problem (4). Then $Z_1^* \leq Z_4^* \leq (1 + \epsilon)Z_1^*$.

Proof. Let x^* be an optimal solution of problem (4). Because the graph of any line $d_i^j + c_i^j x_i^*$ lies on or above the graph of f_i , we know that $f_i(x_i^*) \leq g_i(x_i^*)$ for any $i \in \{1, \dots, n\}$. Therefore, $Z_1^* \leq f(x^*) \leq g(x^*) = Z_4^*$.

Conversely, let x^* be an optimal solution of problem (1) satisfying Assumption 1(b). It suffices to show that $g_i(x_i^*) \leq (1 + \epsilon)f_i(x_i^*)$ for all $i \in \{1, \dots, n\}$. If $x_i^* = 0$, then the inequality holds. Otherwise, let $j = \left\lfloor \log_{1+2\epsilon} \frac{x_i^*}{l_i} \right\rfloor \geq 0$, so that $\frac{x_i^*}{l_i} \in [(1+2\epsilon)^j, (1+2\epsilon)^{j+1}]$. Depending on the position of $\frac{x_i^*}{l_i}$ within this interval, we distinguish two cases.

If $\frac{x_i^*}{l_i} \leq \frac{(1+2\epsilon)^j + (1+2\epsilon)^{j+1}}{2}$, then, because f_i is concave and nondecreasing,

$$g_i(x_i^*) \leq d_i^j + c_i^j x_i^* \leq d_i^j + c_i^j l_i \frac{(1+2\epsilon)^{j+1} + (1+2\epsilon)^j}{2} \quad (5a)$$

$$= d_i^j + c_i^j l_i (1+\epsilon)(1+2\epsilon)^j \leq (1+\epsilon) \left(d_i^j + c_i^j l_i (1+2\epsilon)^j \right) \quad (5b)$$

$$= (1+\epsilon) f_i((1+2\epsilon)^j) \leq (1+\epsilon) f_i(x_i^*). \quad (5c)$$

If $\frac{x_i^*}{l_i} > \frac{(1+2\epsilon)^j + (1+2\epsilon)^{j+1}}{2}$, then, again because f_i is concave and nondecreasing,

$$g_i(x_i^*) \leq g_i(l_i(1+2\epsilon)^{j+1}) = f_i(l_i(1+2\epsilon)^{j+1}) \quad (6a)$$

$$\leq f_i\left(l_i \frac{(1+2\epsilon)^{j+1} + (1+2\epsilon)^j}{2}\right) + f_i\left(l_i \frac{(1+2\epsilon)^{j+1} - (1+2\epsilon)^j}{2}\right) \quad (6b)$$

$$\leq f_i(x_i^*) + f_i(l_i\epsilon(1+2\epsilon)^j) \leq f_i(x_i^*) + \epsilon f_i(l_i(1+2\epsilon)^j) \quad (6c)$$

$$\leq f_i(x_i^*) + \epsilon f_i(x_i^*) = (1+\epsilon)f_i(x_i^*). \quad (6d)$$

Therefore, $Z_4^* \leq g(x^*) \leq (1+\epsilon)f(x^*) = (1+\epsilon)Z_1^*$. \square

The approximation ratio of $1+\epsilon$ is not tight; a more careful analysis ensures an approximation ratio of $\frac{2\epsilon}{2(\sqrt{2\epsilon+1}-1)} \leq 1 + \frac{\epsilon}{2}$, which is tight. Equivalently, instead of deriving a tighter ratio when using $1 + \lceil \log_{1+2\epsilon} \frac{u_i}{l_i} \rceil$ pieces, we can obtain the $1 + \epsilon$ bound using fewer pieces.

Theorem 2 *It is possible to obtain a $1 + \epsilon$ approximation of problem (1) using functions g_i with $1 + \lceil \log_{1+4\epsilon+4\epsilon^2} \frac{u_i}{l_i} \rceil$ pieces each. The ratio is tight for the given number of pieces, if pieces are introduced according to the tangential outer approximation (2).*

Proof. Omitted. \square

We can derive improved bounds on the number of pieces when the functions are known to belong to particular classes (e.g., logarithmic functions), and even better bounds when the functions are known.

Our approach applies to a broader class of problems. Consider the problem $\min\{f(x) : x \in X\}$, with $f : \text{conv}(X) \rightarrow \mathbb{R}$ concave and $f(x) \geq 0, \forall x \in X$. Assume that the problem has an optimal solution x^* and bounds $0 < l_i, u_i$ so that for each $i \in \{1, \dots, n\}$, $|x_i^*| \leq u_i$ and either $f_i(x_i^*) = 0$ or $l_i \leq \min\{|x_i^* - x_i| : f(x_i) = 0\}$. Then we can approximate the problem within a factor of $1 + \epsilon$ by replacing each function f_i with a piecewise linear function g_i with $3 + 2 \lceil \log_{1+4\epsilon+4\epsilon^2} \frac{2u_i}{l_i} \rceil$ pieces, and at most two discontinuity points at the zeroes of f_i . Details will appear in a full version of this paper.

We conclude this section by noting that, as a function of ϵ , the number of pieces grows as $\frac{1}{\log(1+4\epsilon+4\epsilon^2)}$. Since $\lim_{\epsilon \rightarrow 0} \frac{4\epsilon}{\log(1+4\epsilon+4\epsilon^2)} = 1$, as $\epsilon \rightarrow 0$, the number of pieces behaves as $\frac{1}{4\epsilon}$. This behavior enables us to apply the approximation technique to practical concave cost problems. Before presenting some computational experience, we exploit the logarithmic dependence of our results on u_i and l_i to derive polynomial bounds on the number of pieces for a large class of problems.

3 Polyhedral Ground Sets

Let $f(x) = \sum_{i=1}^n f_i(x_i)$, as before, be a concave cost function, and let $P = \{x : Ax \leq b, x \geq 0\}$ be a rational polyhedron defined by $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$. We consider the problem

$$\min\{f(x) : Ax \leq b, x \geq 0\}. \quad (7)$$

We will bound the optimal solution components in terms of input data size. Following standard practice (see, for example, [7]), we define the size of rational numbers and matrices as the number of bits needed to represent them. More precisely for $r \in \mathbb{Z}$, $\text{size}(r) := 1 + \lceil \log_2(|p| + 1) \rceil$, for $r = \frac{r_1}{r_2} \in \mathbb{Q}$ with $\frac{r_1}{r_2}$ irreducible, $\text{size}(r) := \text{size}(r_1) + \text{size}(r_2)$, and for $A \in \mathbb{Q}^{m \times n}$, $\text{size}(A) := mn + \sum_{i=1}^m \sum_{j=1}^n \text{size}(a_{ij})$. The following property is well-known (see [7] and also [8]).

Lemma 1 *Any vertex x of P has $\text{size}(x) \leq U(A, b) := 4(\text{size}(A) + \text{size}(b) + n^2 + 5n)$.*

To approximate problem (7), we introduce the piecewise linear functions g_i , obtained by setting $1 + p_i = 1 + p = 1 + \left\lceil \frac{2}{\log_2(1+4\epsilon+4\epsilon^2)} U(A, b) \right\rceil$ in equations (2) and (3). Consider the problem

$$\min\{g(x) : Ax \leq b, x \geq 0\}. \quad (8)$$

Theorem 3 *If f is nondecreasing and $f(0) \geq 0$, then $Z_\gamma^* \leq Z_\delta^* \leq (1 + \epsilon)Z_\gamma^*$. Each function g_i has a number of pieces polynomial in $\text{size}(A) + \text{size}(b)$, the input size of problem (7).*

Proof. Because we are minimizing a nondecreasing concave function over $P \subseteq \mathbb{R}_+^n$, problem (7) has an optimal solution x^* at a vertex of P [9]. Lemma 1 ensures that $\text{size}(x_i^*) \leq U(A, b)$ for $i \in \{1, \dots, n\}$. Therefore, $x_i^* \in [2^{-U(A, b)}, 2^{U(A, b)}]$. The approximation property follows from Theorem 1. \square

Again, a generalization is possible. Consider the problem $\min\{f(x) : Ax \leq b\}$, with $P = \{x : Ax \leq b\}$, $f : P \rightarrow \mathbb{R}$ concave and $f(x) \geq 0, \forall x \in P$. Any concave function f_i that is not constant over the feasible polyhedron will have at most two zeroes; denote them by ζ'_i and ζ''_i . If the problem has an optimal solution, we can approximate it within a factor of $1 + \epsilon$ by replacing each function f_i with a piecewise linear function g_i with $3 + 2 \left\lceil \frac{1}{\log_2(1+4\epsilon+4\epsilon^2)} (U(A, b) + \text{size}(\zeta'_i) + \text{size}(\zeta''_i)) \right\rceil$ pieces, and at most two points of discontinuity at ζ'_i and ζ''_i . If ζ'_i and ζ''_i are polynomial in the size of the input (or if they are part of the input), then the number of pieces in each g_i is also polynomial in the size of the input. The details will appear in the full length version of this paper.

3.1 Representing the Piecewise Linear Functions

To solve the problems resulting from our approximation technique, we could use several classical methods for representing piecewise linear functions as mixed integer programs. Two of these methods, the incremental and convex combination

approaches (see [10,1]), introduce one or more binary variables for each piece and add a coupling constraint that ensures the approximation uses only one piece.

However, since the objective function to be minimized is concave, the coupling constraint is unnecessary, and we obtain the following fixed charge formulation for problem (8):

$$\min \sum_{i=1}^n \sum_{j=1}^p (f_i(0) + d_i^j z_i^j + c_i^j y_i^j), \quad (9a)$$

$$\text{s.t. } Ax \leq b, \quad (9b)$$

$$x_i = \sum_{j=1}^{p_i} y_i^j, \quad 1 \leq i \leq n, \quad (9c)$$

$$0 \leq y_i^j \leq u_i z_i^j, \quad 1 \leq i \leq n, 1 \leq j \leq p_i, \quad (9d)$$

$$z_i^j \in \{0, 1\}, \quad 1 \leq i \leq n, 1 \leq j \leq p_i. \quad (9e)$$

Lemma 2 *Problems (8) and (9) are equivalent. The input size of problem (9) is polynomial in the input size of problem (8).*

Proof. Omitted. \square

One key advantage of formulation (9) is that, for many practical problems with special structure, the resulting model has the same special structure, but, instead of general concave costs, it has fixed charge costs. Therefore, solution methods for fixed charge problems are available for approximately solving problems with general concave costs.

While featuring (simpler) fixed charge costs instead of general concave cost functions, and the same number of constraints, problem (9) has $1+p$ times more variables. Although for general polyhedra p could be prohibitively large, for many practical problems, we are able to derive significantly smaller expressions for p and, therefore, solve even large problems.

4 Flows in Networks

To illustrate our approach on a practical problem, we consider uncapacitated multicommodity flows (see [11] for a survey and applications). Let $G = (V, E)$, $|V| = n$, $|E| = m$ be an undirected graph, and consider the problem

$$\min \left\{ \sum_{e \in E} f_e \left(\sum_{k=1}^K x_e^k \right) : \sum_{e \in \delta^+(v)} x_e^k - \sum_{e \in \delta^-(v)} x_e^k = b_v^k, x_e^k \geq 0 \right\}. \quad (10)$$

K is the number of commodities, x_e^k denotes the flow of commodity k on edge e , and b_v^k is the supply ($b_v^k > 0$) or demand ($b_v^k < 0$) of commodity k at node v . Let

$b_{\min} := \min\{|b_v^k| : v \in V, 1 \leq k \leq K\}$, $B^k := \sum_{v:b_v^k > 0} b_v^k$, and $B := \sum_{k=1}^K B^k$. For simplicity, we assume that the problem data are integral, and that $f_e(0) = 0$ for $e \in E$.

In this setting, formulation (9) becomes $\min \left\{ \sum_{e \in E} \sum_{j=1}^p \sum_{k=1}^K (d_e^j z_e^j + c_e^j \times y_e^{k,j}) : \sum_{e \in \delta_G^+(v)} \sum_{j=1}^p y_e^{k,j} - \sum_{e \in \delta_G^-(v)} \sum_{j=1}^p y_e^{k,j} = b_v^k, 0 \leq y_e^{k,j} \leq B^k y_e^j, y_e^j \in \{0, 1\} \right\}$. On the graph $G' = (V, E')$ with $E' = \{e^j : e \in E, 1 \leq j \leq p\}$, this problem becomes again an uncapacitated multicommodity flow problem, but now on a network with $(1 + p)m$ edges.

$$\min \sum_{e \in E'} \sum_{k=1}^K (d_e z_e + c_e x_e^k), \quad (11a)$$

$$\text{s.t. } \sum_{e \in \delta^+(v)} x_e^k - \sum_{e \in \delta^-(v)} x_e^k = b_v^k, \quad v \in V, 1 \leq k \leq K, \quad (11b)$$

$$0 \leq x_e^k \leq z_e, \quad e \in E, 1 \leq k \leq K, \quad (11c)$$

$$z_e \in \{0, 1\}, \quad e \in E. \quad (11d)$$

However, the new problem has fixed charge edge costs instead of general concave costs. This model is the well-known uncapacitated network design problem [12,13,14], with d_e the installation cost of edge e and c_e the cost of routing flow on e once it is installed.

Theorem 4 $Z_{10}^* \leq Z_{11}^* \leq (1 + \epsilon)Z_{10}^*$. This ratio can be achieved using $1 + \lceil \log_{1+4\epsilon+4\epsilon^2} \frac{B}{b_{\min}} \rceil \leq 1 + \lceil \log_{1+4\epsilon+4\epsilon^2} B \rceil$ pieces for each edge cost function.

Proof. As is well-known [9,15], in some optimal solution to problem (10), the flow for each commodity occurs on a tree. Consequently, any nonzero flow on any edge will be at least $b_{\min} \geq 1$. The approximation result now follows from Theorem 3 and Lemma 2. \square

The special structure of the problem allows us to increase the number of edges by a factor of only $1 + \lceil \log_{1+4\epsilon+4\epsilon^2} B \rceil$, which is much less than the factor obtained for general polyhedra.

4.1 Computational Results

We present preliminary computational results for uncapacitated multicommodity flow problems with complete uniform demand of 1 unit. Using the dual ascent method described by Balakrishnan et al. [6] (also known as the primal-dual method [14]), we have been able to solve fairly large size problems, with up 50 nodes, 500 edges, and 1,250,000 flow variables, to within five to twenty percent of guaranteed optimality (see Table 1). To the best of our knowledge, the literature contains few exact or approximate computational results for general separable concave cost multicommodity network flow problems with full demand; Bell and Lamar [16] examine single-commodity flows on considerably smaller networks.

Table 1. Network sizes and computational results. The values in column DA are the approximation ratios obtained by the dual ascent algorithm for the piecewise linear problems. The values in column Overall are the overall guaranteed approximation ratios, which equal $100 \left(1 - (1 + \epsilon) \left(1 + \frac{DA}{100}\right)\right)$.

#	Original network			Resulting network			Computation		
	n	m	Variables	100ϵ	m'	Variables	Time	DA	Overall
1	10	20	2,000	5%	420	42,000	0.16s	0%	5%
2	15	50	11,250	5%	1,300	292,500	2.57s	2.64%	7.76%
3	20	100	40,000	5%	2,900	1,160,000	34.6s	5.29%	10.6%
4	30	120	108,000	5%	3,960	3,564,000	59.22s	8.52%	13.9%
5	30	300	270,000	5%	9,900	8,910,500	10m18s	10.7%	16.2%
6	40	160	256,000	5%	5,760	9,216,000	5m36s	3.54%	8.71%
7	40	400	640,000	10%	8,000	12,800,000	6m4s	0.49%	10.5%
8	50	200	500,000	5%	7,800	19,500,000	30m47s	9.48%	14.9%
9	50	500	1,250,000	10%	10,500	26,250,000	61m2s	8.6%	19.5%

We have performed all the computations on a Pentium IV Xeon 2.4Ghz with 1GB of RAM, using randomly generated problems. To ensure feasibility, for each problem we first generated a random spanning tree. Then we added the desired number of edges between nodes selected uniformly at random. We chose the cost function f_e for $e \in E$ as follows. Set $f_e(0) = 0$. For $x_e > 0$, choose $f_e(x_e)$, with equal probability, as either $f_e(x_e) = a + b \log(x_e)$ or $f_e(x_e) = a + b(x_e)^c$, and then generate a, b and, if required, c at random, from uniform distributions over $[\frac{1}{10}, 10]$, $[\frac{1}{3}, \frac{100}{3}]$, and $[\frac{1}{101}, \frac{100}{101}]$. The choice of ϵ for these computations is limited by the size problem that the current dual ascent code is able to solve.

All the error bounds in Table 1 are worst case bounds; the actual bounds are likely to be smaller, perhaps considerably smaller. For example, by substituting the solution for problem 1 into the concave cost function, we obtain a value within 2% of the piecewise cost, and after solving problem 2 with an MIP solver, we found that the dual ascent generated IP solution was 0.09% (not 2.64%) from optimality. More extensive computational results, that take advantage of the ability of dual ascent to handle even larger problems, sparse demand patterns, and non-uniform demand, and that use a more refined assessment of the error bounds, will be available in a full version of this paper.

Acknowledgments

This research was partially supported by the Singapore-MIT Alliance (SMA). We are grateful to Professor Anant Balakrishnan for providing us with the original version of the dual ascent code.

References

1. Nemhauser, G., Wolsey, L.: Integer and combinatorial optimization. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., New York (1999) Reprint of the 1988 original, A Wiley-Interscience Publication.
2. Meyerson, A., Munagala, K., Plotkin, S.: Cost-distance: two metric network design. In: 41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000). IEEE Comput. Soc. Press, Los Alamitos, CA (2000) 624–630
3. Munagala, K.: Approximation algorithms for concave cost network flow problems. PhD thesis, Stanford University, Department of Computer Science (2003)
4. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows. Prentice Hall Inc., Englewood Cliffs, NJ (1993) Theory, algorithms, and applications.
5. Hochbaum, D.S., Shanthikumar, J.G.: Convex separable optimization is not much harder than linear optimization. *J. Assoc. Comput. Mach.* **37** (1990) 843–862
6. Balakrishnan, A., Magnanti, T.L., Wong, R.T.: A dual-ascent procedure for large-scale uncapacitated network design. *Oper. Res.* **37** (1989) 716–740
7. Korte, B., Vygen, J.: Combinatorial optimization. Second edn. Volume 21 of Algorithms and Combinatorics. Springer-Verlag, Berlin (2002) Theory and algorithms.
8. Grötschel, M., Lovász, L., Schrijver, A.: Geometric algorithms and combinatorial optimization. Second edn. Volume 2 of Algorithms and Combinatorics. Springer-Verlag, Berlin (1993)
9. Bauer, H.: Minimalstellen von Funktionen und Extrempunkte. *Arch. Math.* **9** (1958) 389–393
10. Croxton, K.L., Gendron, B., Magnanti, T.L.: A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science* **49** (2003) 1268–1273
11. Guisewite, G.M., Pardalos, P.M.: Minimum concave-cost network flow problems: applications, complexity, and algorithms. *Ann. Oper. Res.* **25** (1990) 75–99 Computational methods in global optimization.
12. Ball, M.O., Magnanti, T.L., Monma, C.L., eds.: Network models. Volume 7 of Handbooks in Operations Research and Management Science. North-Holland Publishing Co., Amsterdam (1995)
13. Balakrishnan, A., Magnanti, T.L., Mirchandani, P.: Network design. In Dell'Amico, M., Maffioli, F., eds.: Annotated bibliographies in combinatorial optimization. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Ltd., Chichester (1997) 311–334 A Wiley-Interscience Publication.
14. Goemans, M., Williamson, D.: The primal-dual method for approximation algorithms and its application to network design problems. In Hochbaum, D.S., ed.: Approximation algorithms for NP-hard problems. PWS Pub. Co., Boston (1997) 144–191
15. Schrijver, A.: Combinatorial optimization. Polyhedra and efficiency. Vol. A. Volume 24 of Algorithms and Combinatorics. Springer-Verlag, Berlin (2003) Paths, flows, matchings, Chapters 1–38.
16. Bell, G.J., Lamar, B.W.: Solution methods for nonconvex network flow problems. In: Network optimization (Gainesville, FL, 1996). Volume 450 of Lecture Notes in Econom. and Math. Systems. Springer, Berlin (1997) 32–50

Three Kinds of Integer Programming Algorithms Based on Barvinok's Rational Functions

J.A. De Loera, D. Haws, R. Hemmecke, P. Huggins, and R. Yoshida

Dept. of Mathematics,
University of California,
Davis CA 95616, USA,
`latte@math.ucdavis.edu`,
<http://www.math.ucdavis.edu/~latte>

Abstract. This paper presents three kinds of algebraic-analytic algorithms for solving integer and mixed integer programming problems. We report both theoretical and experimental results. We use the generating function techniques introduced by A. Barvinok.

1 Introduction

In 1993 A. Barvinok gave an algorithm that *counts* lattice points in convex rational polyhedra in polynomial time when the dimension of the polytope is fixed (see [3,4,5] and the references within). Originally, Barvinok's counting algorithm relied on H. Lenstra's polynomial time algorithm for integer programming in a fixed number of variables [13], but shortly after Barvinok's breakthrough, Dyer and Kannan [10] showed that this step can be replaced by a short-vector computation using the *LLL* algorithm. Therefore, using binary search, one can turn Barvinok's counting oracle into an algorithm that solves integer programming problems with a fixed number of variables in polynomial time (i.e. by counting the number of lattice points in P that satisfy $c \cdot x \geq \alpha$, we can narrow the range for the maximum value of $c \cdot x$, then we iteratively look for the largest α where the count is non-zero). This idea was proposed by Barvinok in [4]. We call this IP algorithm the *BBS algorithm*.

More recently, J.B. Lasserre outlined an asymptotic heuristic method for solving integer programs [12], or at least providing an upper bound on the optimal value, which is also based on Barvinok's rational functions. Unfortunately, Lasserre's criteria, needed to find an optimum value, often fail to hold in practice. Within this topic we make three contributions:

1. In Section 2 we present yet another way to use Barvinok's rational functions to solve integer programs, the *(A, b, c)-test set algorithm*. This time the rational functions encode a *test set*. A test set for a family of integer programs is a finite collection of integral vectors with the property that every feasible non-optimal solution of any integer program in the family can be improved by adding a vector in the test set. There has been considerable activity in the

area of test sets and augmentation methods (e.g. Graver and Gröbner bases, integral basis method, etc.) [2,15]. Here we promote a new way of looking at test sets, as explained in [8,11]: First, encode the whole test set in a “short” sum of rational functions. Second, instead of repeatedly improving one step at a time, the optimum can be obtained via a single Hadamard product of two rational functions. We obtain the following complexity results:

Theorem 1. *Let $A \in \mathbb{Z}^{m \times d}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^d$, $J \subset \{1, \dots, d\}$, and assume that the number of variables d is fixed. Suppose $P := \{x \in \mathbb{R}^d : Ax \leq b, x \geq 0\}$ is a rational convex polytope in \mathbb{R}^d . Given the mixed-integer programming problem*

$$\text{maximize } c \cdot x \text{ subject to } x \in \{x \in \mathbb{R}^d : x \in P, x_i \in \mathbb{Z} \text{ for } i \in J \subset \{1, \dots, d\}\},$$

(A) *We can use rational functions to encode the set of vectors (the (A, b, c) -test set):*

$\{u - v : u$ is a c -optimal solution, v feasible solution, $u, v \in \mathbb{Z}^d\}$, and then solve the MIP problem in time polynomial in the size of the input.

(B) *More strongly, the (A, b, c) test set can be replaced by smaller test sets, such as Graver bases or reduced Gröbner bases.*

2. We improve Lasserre’s heuristic and give a third kind of deterministic IP algorithm based on Barvinok’s rational function algorithms, the *digging algorithm*. In this case the algorithm can have an exponential number of steps even for fixed dimension, but performs well in practice. See Section 3 for details.
3. We implemented the *BBS algorithm* and the *digging algorithm* in the second release of the computer software **LattE** (see [7,8,9]). We solved several challenging knapsack problems and compared the performance of **LattE** with the mixed-integer programming solver CPLEX version 6.6. In fact the digging algorithm is often surpassed by what we call the *single cone digging algorithm*. See Section 4 for computational tests.

2 The (A, b, c) Test Set Algorithm

In all our discussions below, the input data are an $m \times d$ integral matrix A and an integral m -vector b . For simplicity we assume it describes a nonempty full dimensional convex polytope $P = \{x \in \mathbb{R}^d : Ax \leq b, x \geq 0\}$. Our point of view is that lattice points will be encoded as the exponent vectors of monomials. For example, $(2, -11)$ is represented by $z_1^2 z_2^{-11}$. The set of all lattice points in the polytope P will be represented by the multivariate generating function $f(P; z) = \sum_{a \in P \cap \mathbb{Z}^d} z^a$, (where $z^a := z_1^{a_1} z_2^{a_2} \dots z_d^{a_d}$) with one monomial in the sum per lattice point inside P . The crucial fact from Barvinok’s theory is that the exponentially large sum $f(P; z)$ can be written as a short sum of rational functions of the form

$$f(P; z) = \sum_{i \in I} E_i \frac{z^{u_i}}{\prod_{j=1}^d (1 - z^{v_{ij}})}, \quad (1)$$

where I is a polynomial-size indexing set, and where $E_i \in \{1, -1\}$ and $u_i, v_{ij} \in \mathbb{Z}^d$ for all i and j . More recently Barvinok and Woods (2003) further showed how to carry out Boolean operations with sets of lattice points when they are encoded as rational functions:

Lemma 1 (Theorem 3.6 in [5]).

Let S_1, S_2 be finite subsets of \mathbb{Z}^d , for d fixed. Let $f(S_1; x)$ and $f(S_2; x)$ be their generating functions, given as short rational functions with at most k binomials in each denominator. Then there exists a polynomial time algorithm, which, given $f(S_i, x)$, computes

$$f(S_1 \cap S_2; x) = \sum_{i \in I} \gamma_i \cdot \frac{x^{u_i}}{(1 - x^{v_{i1}}) \dots (1 - x^{v_{is}})}$$

with $s \leq 2k$, where the γ_i are rational numbers and u_i, v_{ij} are nonzero integral vectors.

We will use this *Intersection Lemma* to extract special monomials present in the expansion of a generating function. The essential step in the intersection algorithm is the use of the *Hadamard product* [5, Definition 3.2] and a special monomial substitution. The Hadamard product is a bilinear operation on rational functions (we denote it by $*$). The computation is carried out for pairs of summands as in [5]. Note that the Hadamard product $m_1 * m_2$ of two monomials m_1, m_2 is zero unless $m_1 = m_2$. Another key subroutine introduced by Barvinok and Woods is the following *Projection Theorem*.

Lemma 2 (Theorem 1.7 in [5]).

Assume the dimension d is a fixed constant. Consider a rational polytope $P \subset \mathbb{R}^d$ and a linear map $T : \mathbb{Z}^d \rightarrow \mathbb{Z}^k$. There is a polynomial time algorithm which computes a short representation of the generating function $f(T(P \cap \mathbb{Z}^d); z)$.

To extract an explicit optimal solution we need the following lemma.

Lemma 3 (Lemma 8 in [8]). Assume the dimension d is fixed. Let $S \subset \mathbb{Z}_+^d$ be nonempty and finite. Suppose the polynomial $f(S; z) = \sum_{\beta \in S} z^\beta$ is represented as a short rational function and let c be a cost vector. We can extract the (unique) lexicographic largest leading monomial from the set $\{x^\alpha : \alpha \cdot c = M, \alpha \in S\}$, where $M := \max\{\alpha \cdot c : \alpha \in S\}$, in polynomial time.

Proof of Theorem 1: For lack of space, we only show the proof of part (A). For the details of part (B) see [8]. We first explain how to solve integer programs (where all variables are demanded to be integral). This part of the proof is essentially the proof of Lemma 3.1 given in [11] for the case $Ax = b$, $x \geq 0$, instead of $Ax \leq b$,

but we emphasize the fact that b is fixed here. We will see how the techniques can be extended to mixed integer programs later. For a positive integer R , let

$$r(x, R) = \prod_{i=1}^n \left(\frac{1}{1-x_i} + \frac{x_i^R}{1-x_i^{-1}} \right)$$

be the generating function encoding all x -monomials in the positive orthant, having degree at most R in any variable. Note that this is a large, but finite, set of monomials. Suppose $P = \{x \in \mathbb{R}^d : Ax \leq b, x \geq 0\}$ is a nonempty polytope. Using Barvinok's algorithm in [4], compute the following generating function in $2d$ variables:

$$f(x, y) = \sum \{ x^u y^v : Au \leq b, Av \leq b, u, v \geq 0, c \cdot u - c \cdot v \geq 1, \text{ and } u, v \in \mathbb{Z}^d \}.$$

This is possible because we are clearly dealing with the lattice points of a rational polytope. The monomial expansion of $f(x, y)$ exhibits a clear order on the variables: $x^u y^v$ where $c \cdot u > c \cdot v$. Hence v is not an optimal solution. In fact, optimal solutions will never appear as exponents in the y variables.

Now let $g(y)$ be the projection of $f(x, y)$ onto the y -variables variables. Thus $g(y)$ is encoding all non-optimal feasible integral vectors (because the exponent vectors of the x 's are better feasible solutions, by construction), and it can be computed from $f(x, y)$ in polynomial time by Lemma 2. Let $V(P)$ be the vertex set of P and choose an integer $R \geq \max\{v_i : v \in V(P), 1 \leq i \leq d\}$ (we can find such an integer R via linear programming). Define $f(x, y)$ and $g(x)$ as above and compute the Hadamard product

$$H(x, y) := f(x, y) * [(r(x, R) - g(x)) r(y, R)].$$

This is the sum over all monomials $x^u y^v$ where $u, v \in P$ and where u is an optimal solution. The reader should note that the vectors $u - v$ form a test set (an enormous one), since they can be used to improve from any feasible non-optimal solution v . This set is what we called the (A, b, c) -test set. It should be noted that, with more work, one may replace $H(x, y)$ by a similar encoding of other test sets, such as the Graver test set or a Gröbner basis (see [8] for details).

We now use $H(x, y)$ as one would use a traditional test set for finding an optimal solution: Find a feasible solution a inside the polytope P using Lemma 3 and Barvinok's Equation (1). Improve or augment to an optimal solution by computing the Hadamard product

$$H(x, y) * (y^a r(x, R)).$$

The result is the set of monomials of the form $x^u y^a$ where u is an optimal solution. One monomial of the set, say the lexicographic largest, can be obtained by applying Lemma 3. This concludes the proof of the case when all variables are integral.

Now we look at the mixed integer programming case, where only x_i with $i \in J \subset \{1, \dots, d\}$ are required to be integral. Without loss of generality, we

may assume that $J = \{r, \dots, d\}$ for some r , $1 \leq r \leq d$. Thus, splitting A into $(B|C)$, we may write the polytope P as $\{(x, x') : Bx + Cx' \leq b, x, x' \geq 0\}$ where the variables corresponding to B are not demanded to be integral. Consider a vertex optimal solution \bar{x} to the mixed integer problem. The first key observation is that its fractional part can be written as $\bar{x}_J = \hat{B}^{-1}(b - C\bar{x}')$ where $b - C\bar{x}'$ is an integer vector. Here \hat{B}^{-1} denotes the inverse of a submatrix of B . This follows from the theory of linear programming, when we solve the mixed integer program for fixed $x' = \bar{x}'$.

The denominators appearing are then contributed by \hat{B}^{-1} . Then every appearing denominator is a factor of M , the least common multiple of all determinants of a square submatrix of A . It is clear M can be computed in polynomial time in the size of the input. This complexity bound holds, since the number of such square submatrices is bounded by a polynomial in m , the number of rows of A , of degree d , the number of columns of A . Moreover, each of these determinants can be computed in time polynomial in the size of the input, and therefore, M itself can be computed in time polynomial in the size of the input in fixed dimension d . Thanks to this information, we know that if we dilate the original polytope P by M , the optimal solutions of the mixed integer program become, in the dilation MP , optimal integral solutions of the problem

$$\text{maximize } c \cdot x \text{ subject to } x \in MP, x \in \mathbb{Z}^d$$

with the additional condition that the coordinates with index in J are multiples of M . Ignoring this condition involving multiples of M for a moment, we see that, as we did before, we can obtain an encoding of *all* optimal improvements as a generating function $H(x, y)$.

$$\text{Let } \bar{r}(x, R) = \left[\prod_{i \notin J} \left(\frac{1}{1-x_i} + \frac{x_i^R}{1-x_i^{-1}} \right) \right] \left[\prod_{i \in J} \left(\frac{1}{1-x_i^M} + \frac{x_i^{RM}}{1-x_i^{-M}} \right) \right].$$

To extract those vectors whose coordinates indexed by J are multiples of M , we only need to intersect (Hadamard product again) our generating function $H(x, y)$ with the generating function $\bar{r}(x, R)\bar{r}(y, R)$. Then only those vectors whose coordinates indexed by J are multiples of M remain. This completes the proof of the theorem. \square

3 The Digging Algorithm

In what follows we present a strengthening of Lasserre's heuristic and discuss how to use Barvinok's short rational functions to solve integer programs using digging. Given $A \in \mathbb{Z}^{m \times d}$, $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}^d$, we consider the integer programming problem of the form $\text{maximize } \{c \cdot x : Ax \leq b, x \geq 0, x \in \mathbb{Z}^d\}$. We assume that the input system of inequalities $Ax \leq b, x \geq 0$ defines a polytope $P \subset \mathbb{R}^d$, such that $P \cap \mathbb{Z}^d$ is nonempty. As before, all integer points are encoded as a short rational function $f(P; z)$ in Equation (1) for P , where the rational function is given in Barvinok's form. Remember that if we were to expand Equation (1) into monomials (generally a very bad idea!) we would get $f(P; z) = \sum_{\alpha \in P \cap \mathbb{Z}^d} z^\alpha$. For

a given $c \in \mathbb{Z}^d$, if we make the substitution $z_i = t^{c_i}$, Equation (1) yields a univariate rational function in t :

$$f(P; t) = \sum_{i \in I} E_i \frac{t^{c \cdot u_i}}{\prod_{j=1}^d (1 - t^{c \cdot v_{ij}})}. \quad (2)$$

The key observation is that if we make that substitution directly into the monomial expansion of $f(P; z)$, we have that the multivariate monomial becomes $z^\alpha \rightarrow t^{c \cdot \alpha}$. Moreover we would obtain the relation

$$f(P; t) = \sum_{\alpha \in P \cap \mathbb{Z}^d} t^{c \cdot \alpha} = kt^M + (\text{lower degree terms}), \quad (3)$$

where M is the optimal value of our integer program and where k counts the number of optimal integer solutions. Unfortunately, in practice, M and the number of lattice points in P may be huge and we need to avoid the monomial expansion step altogether. All computations have to be done by manipulating short rational functions in (2).

Lasserre [12] suggested the following approach: For $i \in I$, define sets η_i by $\eta_i = \{j \in \{1, \dots, d\} : c \cdot v_{ij} > 0\}$, and define vectors w_i by $w_i = u_i - \sum_{j \in \eta_i} v_{ij}$. Let n_i denote the cardinality of η_i . Now define $M = \max\{c \cdot w_i : i \in I\}$, $S = \{i \in I : c \cdot w_i = M\}$ and set $\sigma = \sum_{i \in S} E_i (-1)^{n_i}$. Note that M simply denotes the highest exponent of t appearing in the expansions of the rational functions defined for each $i \in I$ in (2). The number σ is in fact the sum of the coefficients of t^M in these expressions, that is, σ is the coefficient of t^M in $f(P; t)$. Now with these definitions and notation we can state the following result proved by Lasserre [12].

Theorem 2 (Theorem 3.1 in [12]). *If $c \cdot v_{ij} \neq 0$ for all $i \in I, j \in \{1, \dots, d\}$, and if $\sigma \neq 0$, then M is the optimal value π of the integer program maximize $\{c \cdot x : Ax \leq b, x \geq 0, x \in \mathbb{Z}^d\}$.*

When the hypotheses of Theorem 2 are met, from an easy inspection, we could recover the optimal value of an integer program. If we assume that c is chosen randomly from some large cube in \mathbb{Z}^d , then the first condition is easy to obtain. Unfortunately, our computational experiments (see Section 4) indicate that the condition $\sigma \neq 0$ is satisfied only occasionally. Thus an improvement on the approach that Lasserre proposed is needed to make the heuristic terminate in all instances. Here we explain the details of an algorithm that *diggs* for the coefficient of the next highest appearing exponent of t . For simplicity our explanation assumes the easy-to-achieve condition $c \cdot v_{ij} \neq 0$, for all v_{ij} .

As before, take Equation (1) computed via Barvinok's algorithm. Now, for the given c , we make the substitutions $z_k = y_k t^{c_k}$, for $k = 1, \dots, d$. Then substitution into (1) yields a sum of multivariate rational functions in the vector variable y and scalar variable t :

$$g(P; y, t) = \sum_{i \in I} E_i \frac{y^{u_i} t^{c \cdot u_i}}{\prod_{j=1}^d (1 - y^{v_{ij}} t^{c \cdot v_{ij}})}. \quad (4)$$

On the other hand, the substitution on the left-side of Equation (1) gives the following sum of monomials.

$$g(P; y, t) = \sum_{\alpha \in P \cap \mathbb{Z}^d} y^\alpha t^{c \cdot \alpha} = \sum_{n=M}^{-\infty} \left(\sum_{\alpha \in P \cap \mathbb{Z}^d, \alpha \cdot c = n} y^\alpha \right) t^n. \quad (5)$$

Both equations, (5) and (4), represent the same function $g(P; y, t)$. Thus, if we compute a Laurent series of (4) that shares a region of convergence with the series in (5), then the corresponding coefficients of both series must be equal. In particular, because P is a polytope, the series in (5) converges almost everywhere. Thus if we compute a Laurent series of (4) that has any nonempty region of convergence, then the corresponding coefficients of both series must be equal. Barvinok's algorithm provides us with the right hand side of (4). We need to obtain the coefficient of highest degree in t from the expanded Equation (refeq:d). We compute a Laurent series for it using the following procedure: Apply the identity

$$\frac{1}{1 - y^{v_{ij}} t^{c \cdot v_{ij}}} = \frac{-y^{-v_{ij}} t^{-c \cdot v_{ij}}}{1 - y^{-v_{ij}} t^{-c \cdot v_{ij}}} \quad (6)$$

to Equation (4), so that any v_{ij} such that $c \cdot v_{ij} > 0$ can be changed in “sign” to be sure that, for all v_{ij} in (4), $c \cdot v_{ij} < 0$ is satisfied (we may have to change some of the E_i , u_i and v_{ij} using our identity, but we abuse notation and still refer to the new signs as E_i and the new numerator vectors as u_i and the new denominator vectors as v_{ij}). Then, for each of the rational functions in the sum of Equation (4) compute a Laurent series of the form

$$E_i y^{u_i} t^{c \cdot u_i} \prod_{j=1}^d (1 + y^{v_{ij}} t^{c \cdot v_{ij}} + (y^{v_{ij}} t^{c \cdot v_{ij}})^2 + \dots). \quad (7)$$

Multiply out each such product of series and add the resultant series. This yields precisely the Laurent series in (5). Thus, we have now the steps of an algorithm to solve integer programs:

Algorithm: (*Digging Algorithm*):

Input: $A \in \mathbb{Z}^{m \times d}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^d$.

Output: The optimal value and an optimal solution of maximize $\{c \cdot x : Ax \leq b, x \geq 0, x \in \mathbb{Z}^d\}$.

1. Using Barvinok's algorithm compute the rational function expression of Equation (4). Use the identity (6) as necessary to enforce that all v_{ij} in (4) satisfy $c \cdot v_{ij} < 0$.
2. Via the expansion formulas of (7), multiply out the factors and add the terms, grouping together those of the same degree in t . Thus we find (5) by calculating the terms' coefficients. Proceed in decreasing order with respect to the degree of t . This can be done because, for each series appearing in the expansion formulas (7), all $c \cdot v_{ij}$ are negative, so that the terms of the series are given in decreasing order with respect to the degree of t .

3. Continue calculating the terms of the expansion (5), in decreasing order with respect to the degree of t , until a degree n of t is found such that for some $\alpha \in \mathbb{Z}^d$, the coefficient of $y^\alpha t^n$ is non-zero in the expansion (5).
4. Return “ n ” as the optimal value of the integer program and return α as an optimal solution.

We end our discussion by noting one nice feature of the digging algorithm: If one needs to solve a family of integer programs where only the cost vector c is changing, then Equation (4) can be computed once and then apply the steps of the algorithm above for each cost vector to obtain all the optimal values.

Given the polytope $P := \{x \in \mathbb{R}^d : Ax \leq b, x \geq 0\}$, the *tangent cone* K_v at a vertex v of P is the pointed polyhedral cone defined by the inequalities of P that turn into equalities when evaluated at v . We observed in [7] that a major practical bottleneck of the original Barvinok algorithm in [3] is the fact that a polytope may have too many vertices. Since originally one visits each vertex to compute a rational function at each tangent cone, the result can be costly. Therefore a natural idea for improving the digging algorithm is to compute with a single tangent cone of the polytope and revisit the above calculation for a smaller sum of rational functions. Let vertex v^* give the optimal value for the given linear programming problem and we only deal with the tangent cone K_{v^*} . Suppose we have the following integer programming problem:

$$(IP) \text{ maximize } c \cdot x \text{ subject to } x \in P \cap \mathbb{Z}^d,$$

where $P := \{x \in \mathbb{R}^d : Ax \leq b, x \geq 0\}$, $A \in \mathbb{Z}^{m \times d}$ and $b \in \mathbb{Z}^m$.

Then we have the following linear programming relaxation problem for the given integer programming problem:

$$(LP) \text{ maximize } c \cdot x \text{ subject to } x \in P.$$

One of the vertices of P gives the optimal value for (LP) [14]. Let $V(P)$ be the vertex set of P and $v \in V(P)$ be a vertex such that $c \cdot v$ is the optimal value for (LP). Then, clearly, the tangent cone K_v at v contains P . So, if we can find an integral point $x^* \in K_v$ such that $c \cdot x^* \geq c \cdot x$, $\forall x \in P$ and $x^* \in P \cap \mathbb{Z}^d$, then x^* is an optimal solution for (IP). Note that the formulas of rational functions that we used for the original digging algorithm still hold for the rational functions of the tangent cone K_v . The outline for the single cone digging algorithm is the following:

Algorithm: (*Single Cone Digging Algorithm*):

Input: $A \in \mathbb{Z}^{m \times d}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^d$.

Output: optimal value and optimal solution of maximize $\{c \cdot x : Ax \leq b, x \geq 0, x \in \mathbb{Z}^d\}$.

1. Compute a vertex v of P such that $c \cdot v = \max \{c \cdot x : Ax \leq b, x \geq 0\}$.
2. Compute the tangent cone K_v at v and compute the short rational function (4) encoding the lattice points inside K_v .
3. Use the identity (6) as necessary to enforce that all v_{ij} in (4) satisfy $c \cdot v_{ij} < 0$.

Problem	a	b
cuww1	12223 12224 36674 61119 85569	89643482
cuww2	12228 36679 36682 48908 61139 73365	89716839
cuww3	12137 24269 36405 36407 48545 60683	58925135
cuww4	13211 13212 39638 52844 66060 79268 92482	104723596
cuww5	13429 26850 26855 40280 40281 53711 53714 67141	45094584
prob1	25067 49300 49717 62124 87608 88025 113673 119169	33367336
prob2	11948 23330 30635 44197 92754 123389 136951 140745	14215207
prob3	39559 61679 79625 99658 133404 137071 159757 173977	58424800
prob4	48709 55893 62177 65919 86271 87692 102881 109765	60575666
prob5	28637 48198 80330 91980 102221 135518 165564 176049	62442885
prob6	20601 40429 40429 45415 53725 61919 64470 69340 78539 95043	22382775
prob7	18902 26720 34538 34868 49201 49531 65167 66800 84069 137179	27267752
prob8	17035 45529 48317 48506 86120 100178 112464 115819 125128 129688	21733991
prob9	3719 20289 29067 60517 64354 65633 76969 102024 106036 119930	13385100
prob10	45276 70778 86911 92634 97839 125941 134269 141033 147279 153525	106925262

Table 1. knapsack problems.

4. Via the expansion formulas (7), find (5) by calculating the terms' coefficients. Proceed in decreasing order with respect to the degree of t . This can be done because, for each series appearing in the expansion formulas (7), the terms of the series are given in decreasing order with respect to the degree of t .
5. Continue calculating the terms of the expansion (5), in decreasing order with respect to the degree of t , until a degree n of t is found such that:
 - for some $\alpha \in \mathbb{Z}^d$, the coefficient of $y^\alpha t^n$ is non-zero in the expansion (5),
 - $A\alpha \leq b$, $\alpha \geq 0$.
6. Return " n " as the optimal value of the integer program and return α as an optimal solution.

From Table 2 and Table 3, one can see that the single cone digging algorithm is faster compared to the BBS algorithm and the original digging algorithm. This algorithm is also more memory efficient than the original digging algorithm, since the number of unimodular cones for the single cone digging algorithm is much less than the number of unimodular cones for the original digging algorithm.

4 Computational Experiments

In this section we report our experience solving hard knapsack problems from [1,6]. See Table 1 for the data used here. Their form is maximize $c \cdot x$ subject to $ax = b$, $x \geq 0$, $x \in \mathbb{Z}^d$, where $b \in \mathbb{Z}$ and where $a \in \mathbb{Z}^d$ with $\gcd(a_1, \dots, a_d) = 1$. For the cost vector c , we choose the first d components of the vector

$$(213, -1928, -11111, -2345, 9123, -12834, -123, 122331, 0, 0).$$

Problem	Value	Solution	Runtime for Digging (Original)	Runtime for Digging (S. Cone)	Runtime for BBS	Runtime for CPLEX 6.6
cuww1	1562142	[7334 0 0 0 0]	0.4 sec.	0.17 sec.	414 sec.	> 1.5h (OM)
cuww2	-4713321	[3 2445 0 0 0 0]	> 3.5h	> 3.5h	6,600 sec.	> 0.75h (OM)
cuww3	1034115	[4855 0 0 0 0 0]	1.4 sec.	0.24 sec.	6,126 sec.	> 0.75h (OM)
cuww4	-29355262	[0 0 2642 0 0 0 0]	> 1.5h	> 1.5h	38,511 sec.	> 0.75h (OM)
cuww5	-3246082	[1 1678 1 0 0 0 0 0]	> 1.5h	147.63 sec.	> 80h	> 0.75h (OM)
prob1	9257735	[966 5 0 0 1 0 0 74]	51.4 sec.	18.55 sec.	> 3h	> 1h (OM)
prob2	3471390	[853 2 0 4 0 0 0 27]	24.8 sec.	6.07 sec.	> 10h	> 0.75h (OM)
prob3	21291722	[708 0 2 0 0 0 1 173]	48.2 sec.	9.03 sec.	> 12h	> 1.5h (OM)
prob4	6765166	[1113 0 7 0 0 0 0 54]	34.2 sec.	9.61 sec.	> 5h	> 1.5h (OM)
prob5	12903963	[1540 1 2 0 0 0 0 103]	34.5 sec.	9.94 sec.	> 5h	> 1.5h (OM)
prob6	2645069	[1012 1 0 1 0 1 0 20 0 0]	143.2 sec.	19.21 sec.	> 4h	> 2h (OM)
prob7	22915859	[782 1 0 1 0 0 0 186 0 0]	142.3 sec.	12.84 sec.	> 4h	> 1h (OM)
prob8	3546296	[1 385 0 1 1 0 0 35 0 0]	469.9 sec.	49.21 sec.	> 3.5h	> 2.5h (OM)
prob9	15507976	[31 11 1 1 0 0 0 127 0 0]	1,408.2 sec.	283.34 sec.	> 11h	4.7 sec.
prob10	47946931	[0 705 0 1 1 0 0 403 0 0]	250.6 sec.	29.28 sec.	> 11h	> 1h (OM)

Table 2. Optimal values, optimal solutions, and running times for each problem. OM:= Out of memory.

We compared four algorithms: The algorithm implemented in CPLEX version 6.6, the digging algorithm, the single cone digging algorithm, and the BBS algorithm; the last three algorithms are now implemented in *LattE*. *LattE* is now available to the public at www.math.ucdavis.edu/~latte and it is distributed with a manual.

All computations were done on a 1 GHz Pentium PC running Red Hat Linux. Table 2 provides the optimal values and an optimal solution for each problem. As it turns out, there is exactly one optimal solution for each problem. With one exception, CPLEX 6.6. could not solve the given problems. Note that whenever the digging algorithm found the optimal value, it did so much faster than the BBS algorithm. This is interesting, as the worst-case complexity for the digging algorithm is exponential even for fixed dimension, while the BBS has polynomial complexity in fixed dimension. The original digging algorithm fails to find a solution for problems cuww2, cuww4, and cuww5. What happens is that the expansion step becomes costly when more coefficients have to be computed. In these three examples, we computed coefficients for more than 2,500,000, 400,000, and 100,000 powers of t ; all turning out to be 0. The Digging algorithm is slower than CPLEX in problem prob9 because during the execution of Barvinok's unimodular cone decomposition (see pages 15 and 16 of [4]) more than 160,000 cones are generated, leading to an enormous rational function for $f(P; t)$. Moreover, for prob9 more than 3,500 coefficients turned out to be 0, before a non-zero leading coefficient was detected. Finally, in problems cuww1, cuww3, prob2, prob3, prob4, prob6, and prob8, no digging was necessary at all, that is, Lasserre's condition did not fail here. For all other problems, Lasserre's condition did fail and digging steps were necessary to find the first non-vanishing coefficient in the expansion of $f(P; t)$.

problem	Original Digging (A)	Original Digging (B)	Single Cone Digging (A)	Single Cone Digging (B)
cuww 1	110	0	25	0
cuww 2	386	> 2,500,000	79	> 2,500,000
cuww 3	346	0	49	0
cuww 4	364	> 400,000	51	> 400,000
cuww 5	2,514	> 100,000	453	578,535
prob 1	10,618	74,150	1,665	74,150
prob 2	6,244	0	806	0
prob 3	12,972	0	2,151	0
prob 4	9,732	0	1,367	0
prob 5	8,414	1	2,336	1
prob 6	26,448	5	3,418	5
prob 7	20,192	0	2,015	0
prob 8	62,044	0	6,523	0
prob 9	162,035	3,558	45,017	3,510
prob 10	38,638	256	5,128	256

Table 3. Data for the digging algorithm. $A :=$ number of unimodular cones and $B :=$ number of digging levels

Acknowledgements: We are grateful to K. Aardal, A. Barvinok, J.B. Lasserre and B. Sturmfels for useful discussions. This research was supported by NSF grant DMS-0309694 and NSF VIGRE grant DMS-0135345.

References

1. Aardal, K., Lenstra, A.K., and Lenstra, H.W. Jr. *Hard equality constrained integer knapsacks*. Preliminary version in W.J. Cook and A.S. Schulz (eds.), Integer Programming and Combinatorial Optimization: 9th International IPCO Conference, Lecture Notes in Computer Science vol. 2337, Springer-Verlag, 2002, 350–366.
2. Aardal, K., Weismantel, R., and Wolsey, L. *Non-Standard Approaches to Integer Programming*. Discrete Applied Mathematics 123, 2002, 5–74
3. Barvinok, A.I. *Polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed*. Math of Operations Research 19, 1994, 769–779.
4. Barvinok, A.I. and Pommersheim, J. *An algorithmic theory of lattice points in polyhedra*. In: *New Perspectives in Algebraic Combinatorics* (Berkeley, CA, 1996–1997), 91–147, Math. Sci. Res. Inst. Publ. 38, Cambridge Univ. Press, Cambridge, 1999.
5. Barvinok, A.I. and Woods, K. *Short rational generating functions for lattice point problems*. Available at arXiv.math.CO.0211146. J. Amer. Math. Soc. 16, 2003, 957–979.
6. Cornuéjols, G., Urbaniak, R., Weismantel, R., and Wolsey, L.A. *Decomposition of integer programs and of generating sets*. R. E. Burkard, G. J. Woeginger, eds., *Algorithms–ESA 97*. Lecture Notes in Computer Science 1284, Springer-Verlag, 1997, 92–103.

7. De Loera, J.A., Hemmecke, R., Tauzer, J., and Yoshida, R. *Effective lattice point counting in rational convex polytopes*. To appear in Journal of Symbolic Computation.
8. De Loera, J.A., Haws, D., Hemmecke, R., Huggins, P., Sturmels, B., and Yoshida, R. *Short rational functions for toric algebra and applications*. Available at arXiv.math.CO.0307350. To appear in Journal of Symbolic Computation.
9. De Loera, J.A., Haws, D., Hemmecke, R., Huggins, P., Tauzer, J., Yoshida, R. *A User's Guide for LattE v1.1*, 2003. Software package LattE is available at <http://www.math.ucdavis.edu/~latte/>
10. Dyer, M. and Kannan, R. *On Barvinok's algorithm for counting lattice points in fixed dimension*. Math of Operations Research 22, 1997, 545– 549.
11. Hosten, S. and Sturmels, B. *Computing the integer programming gap*. Available at math arXiv math.OC/0301266, 2003.
12. Lasserre, J.B. *Integer programming, Barvinok's counting algorithm and Gomory relaxations*. Operations Research Letters, 32, N2, 2004, 133 – 137.
13. Lenstra, H.W. *Integer Programming with a fixed number of variables*. Mathematics of Operations Research, 8, 1983, 538–548.
14. Schrijver, A. *Theory of Linear and Integer Programming*. Wiley-Interscience, 1986.
15. Thomas, R. *Algebraic methods in integer programming*. Encyclopedia of Optimization (eds: C. Floudas and P. Pardalos), Kluwer Academic Publishers, Dordrecht, 2001.

The Path-Packing Structure of Graphs[☆]

András Sebő¹^{**} and László Szegő²^{***}

¹ CNRS, Graph Theory and Combinatorial Optimization,
Laboratoire Leibniz-IMAG, 46, Avenue Félix Viallet, 38000 Grenoble, France
Andras.Sebo@imag.fr

² Department of Operations Research, Eötvös University,
Pázmány Péter sétány 1/C, H-1117 Budapest, Hungary
szego@math.elte.hu

Abstract. We prove Edmonds-Gallai type structure theorems for Mader's edge- and vertex-disjoint paths including also capacitated variants, and state a conjecture generalizing Mader's minimax theorems on path packings and Cunningham and Geelen's path-matching theorem.

1 Introduction

Let $G = (V, E)$ be a graph, $\mathcal{T} = \{T_1, \dots, T_k\}$ a family of pairwise disjoint subsets of V , $T := T_1 \cup \dots \cup T_k$. The points in T are called *terminal* points. A \mathcal{T} -path is a path between two terminal points in different members of \mathcal{T} . Let $\mu = \mu(G, \mathcal{T})$ denote the maximum number of (fully vertex-) disjoint \mathcal{T} -paths.

A T -path is a path in G with two different endpoints in T and all other points in $V \setminus T$. In the edge-disjoint case we will consider T -paths, and \mathcal{T} will not be defined. Clearly, a given set of paths in G is a set of pairwise edge-disjoint T -paths if and only if the corresponding paths of the line-graph of G are pairwise vertex-disjoint \mathcal{T} -paths, where $\mathcal{T} = \{T_x : x \in T\}$, $T_x := \{v_e : e \in \delta(x)\}$ ($x \in T$), and v_e denotes the vertex of the line-graph corresponding to the edge e , and $\delta(x)$ denotes the set of edges incident to x , and in general, $\delta(X) = \delta_G(X)$ ($X \subseteq V(G)$) is the set of edges with exactly one endpoint in X , $d(X) := d_G(X) = |\delta_G(X)|$. The maximum number of edge-disjoint T -paths of G will be denoted by $\mu^*(G, T)$.

Furthermore, we will use the following notations. In a graph $G = (V, E)$, $E[X]$ denotes the set of edges spanned by X for a subset $X \subseteq V$, and $G(X)$ the subgraph induced by X ; $G - X$ denotes the graph obtained from G by deleting the vertices of $X \subseteq V$. For a subset $F \subseteq E$ of edges, $G - F$ denotes the graph obtained from G by deleting the edges in F .

The main result of this paper is an Edmonds-Gallai type structure theorem for maximum sets of edge-disjoint T -paths and vertex-disjoint \mathcal{T} -paths. What does this mean?

* Supported by European MCRTN Adonet, Contract Grant No. 504438.

** Research supported by the Région Rhône-Alpes, projet TEMPRA.

*** Supported by the Hungarian National Foundation for Scientific Research Grant, OTKA T037547 and an FKFP grant no. 0143/2001 and by Egerváry Research Group of the Hungarian Academy of Sciences.

A simplest example is accessibility in directed graphs: given a directed graph $G = (V, A)$ and $x_0 \in V$, find a directed path from x_0 to the other vertices of G or provide a certificate that there is none.

Everybody knows that there is no path between x_0 and $x \in V$ if and only if there is a *directed cut separating x_0 and x* , that is, a set $X \subseteq V$, $x_0 \in X$, $x \notin X$ so that no arc of G is leaving X . The X will be called a *directed cut*. Let us point out the following phenomenon.

There exists a unique set X_0 which is a directed cut separating x_0 at the same time from every $x \in V$ for which there is no (x_0, x) -path.

Indeed, there exists an arborescence with root x_0 and with a path from x_0 to all the vertices accessible from x_0 , so that the set X_0 of vertices of this arborescence is a directed cut. This problem is a special case of the more general problem of shortest paths from a fixed vertex x_0 of a weighted directed graph without negative directed circuits, and of the uniquely determined potentials that certify the length of these paths for every vertex.

In order to help the reading we provide a brief introduction to the Edmonds-Gallai theorem. Denote by \bar{G} the graph which arises from G by adding a vertex x_0 and joining x_0 to all the vertices of G .

If M is a maximum matching of G , let \bar{M} arise from M by adding the edges x_0x where x is a vertex not saturated by M . We will call *alternating path* (with respect to \bar{M}) a path joining x_0 to a vertex of G starting with an edge in \bar{M} , and containing alternatively edges of \bar{M} and not in \bar{M} ; if the last edge of the path is in \bar{M} we will say the path is *symmetric*, if not, then we will say it is *asymmetric*.

A set $X \subseteq V(G)$ is called a *Tutte-set*, if there exists a maximum matching which covers X and all components of $G - X$ are fully matched within the component except for at most one vertex. (This means that the minimum attains at X in the Tutte-Berge formula about the maximum size of a matching.) If X is a Tutte-set of G , we will denote by $C(X)$ the union of the even (vertex-cardinality) components of $G - X$, and by $D(X)$ the union of the odd components of $G - X$.

Define now $D(G)$ to be the set of vertices which are endpoints of symmetric paths, and $A(G)$ the set of vertices which are not, but are the endpoints of asymmetric paths. Clearly, $D(G) = \{v \in V(G) : \mu(G - v) = \mu(G)\}$, $A(G)$ is the set of neighbors of vertices in $D(G)$, and $C(G)$ is the rest of the vertices.

Exclusion Lemma for matchings: *If G is a graph, M is a maximum matching and X is a Tutte-set, then*

- (i) *there is no alternating path to the vertices of $C(X)$, that is, $C(X) \subseteq C(G)$,*
- (ii) *there is no symmetric alternating path to the vertices of X , that is, $X \subseteq V(G) \setminus D(G)$.*

The essence of the Edmonds-Gallai theorem is that there is a unique ‘maximal’ Tutte-set in the following sense (compare the result with the containments of the lemma):

Theorem 1. *For any graph G , $X := A(G)$ is a Tutte-set, and $C(X) = C(G)$, $D(X) = D(G)$.*

When the Edmonds-Gallai theorem is stated in articles or books, often many other (in our view less essential) statements are also added. Those can be useful in the context of the various goals, but are merely easy corollaries of the above statement which provides the main content, which is: there is a most exclusive Tutte-set that excludes everything that can be excluded with the Exclusion Lemma.

Let us mention some uses of the Edmonds-Gallai theorem – for the generalizations not many of these have been worked out: Linear hull of matchings, matching lattice; Cunningham's analysis of the 'slither' game; Cornuéjols, Hartvigsen and Pulleyblank's solution of the k -gon-free 2-matching problem [1], or of the problem of packing complete graphs (equivalently edges and triangles), etc. In Lovász and Plummer's book [11] the theorem is applied as a basic and standard tool throughout. It is useful in cases where the *set* of maximum matchings is important.

Let us list some other examples of the same kind of theorem:

A generalization of the Edmonds-Gallai theorem for (f, g) -factors (subgraphs with upper and lower bounds and maybe parity constraints) has been proved by Lovász [9], [10]. The relation of this generalization to the corresponding minimax theorem (of Tutte for f -factors) is the same as the relation of the Berge-Tutte theorem for matchings and of the Edmonds-Gallai theorem. Similarly, the minimax theorems for the Chinese-Postman problem (T -joins) can be sharpened to provide an Edmonds-Gallai type theorem for T -joins [17], which can be further applied to disjoint path and cut packing problems or results on the duality gap or stronger integrality properties in these problems. Recently, Spille and Szegő [19] have developed the corresponding sharpening of the minimax theorems on path-matchings.

The analogous results concerning Mader's path-packings will be worked out in this paper. The main result concerns vertex-disjoint paths (Sect. 4). The result about edge-disjoint paths is a consequence. However, because of its simplicity and for a more gradual understanding of this structure, we will first give a different – much simpler – proof for the edge-disjoint case (Sect. 3).

These structure theorems are usually consequences of the algorithms that solve the problems. There are three problems though for which this use of structure theorems has been reversed. This means that the assertion of the theorems is mechanically converted into an optimality criterion. If the 'canonical dual solution' defined from a list of tentative optimal solutions does not satisfy the criterion, then one of the two following cases can happen: either a new 'optimal solution' can be added to the list, improving the 'canonical dual solution'; or a better solution can be found, showing that the solutions of the list were not optimal. Such a proof does in fact not use the structure theorem but states it as a conjecture, and provides a new, algorithmic proof of it. We refer to such an algorithm under the name *structure algorithm*.

A first structure algorithm has been provided by Lovász [11] for matchings. For (f, g) -factors Lovász worked out the Edmonds-Gallai theorems [11]. The corresponding structure algorithm has been worked out in [16] in the lan-

guage of accessibility from a fix vertex in a bidirected graph. This algorithm contains structure algorithms for (f, g) -factors and other factorizations or orientation problems with upper, lower bound and parity constraints. The third generalization of Lovász's algorithm concerns minimum T -joins in graphs [15].

For Mader's path packings, or path-matchings not only the structure algorithms are missing, but there is no combinatorial algorithm solving these problems at all! The present paper is intended to be a first step in this direction by providing the statement of the structure theorem. The algorithms and the applications of the developed results are yet to be found.

In this paper a path P is considered to be a set of vertices $P \subseteq V(G)$. We will also speak about the edges of P , and we will denote them by $E(P)$. If \mathcal{P} is a family of paths, $E(\mathcal{P}) = \cup_{P \in \mathcal{P}} E(P)$. Therefore 'disjoint' paths mean pairwise vertex-disjoint paths, unless it is specified that the paths are 'edge-disjoint' only.

A *half- T -path* (or *half path*) is a path with at least one endpoint in T . The other endpoint may or may not be in T , and the path may also consist of only one point in T . However, if the path has more than one vertex, the two endpoints must be different.

Both in the edge- and vertex-disjoint case the main property for classifying vertices for the path structure is the following: given a vertex of the graph is it possible to add a half path joining a terminal with the given vertex, so that deleting all edges, resp. vertices of that path, the maximum number of T -paths does not change?

We will say that an edge or vertex is *covered* by a given set \mathcal{P} of paths if it is contained in a path of \mathcal{P} . Otherwise it is *uncovered*.

We finish this introduction by stating Mader's theorems.

First we consider the edge-disjoint case. Let $T := \{t_1, t_2, \dots, t_k\}$. A subpartition $\mathcal{A} := \{A_1, A_2, \dots, A_k\}$ of V is said to be a T -partition if $t_i \in A_i$, ($i = 1, \dots, k$). For $X \subseteq V(G)$, a component K of $G - X$ is said to be *odd* (*even*) if $d_G(K)$ is an odd (even) number.

Theorem 2 (Mader [12]). *The maximum number of edge-disjoint T -paths is equal to the minimum value of*

$$|A_0| + \sum_{C \in \mathcal{C}} \left\lfloor \frac{1}{2} d_G(C) \right\rfloor , \quad (1)$$

taken over all T -partitions $\mathcal{A} := \{A_1, A_2, \dots, A_k\}$, where A_0 is the set of edges whose two endpoints are in A_i and A_j with $i \neq j$; $A := \bigcup_{i=1}^k A_i$, and the elements of \mathcal{C} are the vertex-sets of the components of the graph $G - A$.

A T -partition $\mathcal{A} := \{A_1, A_2, \dots, A_k\}$ is called *optimal* if $\text{val}_{G,T}^*(\mathcal{A}) := |A_0| + \sum_{C \in \mathcal{C}} \lfloor \frac{1}{2} d_G(C) \rfloor$ is minimum among all T -partitions, that is, if it is equal to $\mu^*(G, T)$.

Second we consider the vertex-disjoint case. For a surprisingly short (non-algorithmic) proof, see Schrijver [14].

Theorem 3 (Mader [13]). *The maximum number of disjoint \mathcal{T} -paths is equal to the minimum value of*

$$|U_0| + \sum_{i=1}^n \left\lfloor \frac{1}{2}|B_i| \right\rfloor , \quad (2)$$

taken over all partitions U_0, U_1, \dots, U_n of V such that each \mathcal{T} -path disjoint from U_0 traverses some edge spanned by some U_i . B_i denotes the set of vertices in U_i that belong to T or have a neighbor in $V - (U_0 \cup U_i)$.

We will use the following reformulation of Theorem 3. It has several advantages:

First – and this is crucial for us now – it provides the right context for a structure theorem in the sense that the parts of a dual solution correspond exactly to the partition provided by the structure theorem. We also think that this formulation is more natural in view of the edge-version of Mader's theorem (Theorem 2), because it has the same form and the latter is sitting in it with a pure specialization. Third, it makes explicit what the sentence ‘each \mathcal{T} -path disjoint from U_0 traverses some edge spanned by some U_i ’ means in Theorem 3. Indeed, the meaning of this sentence is that the vertices of U_0 and the edges spanned by the U_i ($i = 1, \dots, k$) block all the \mathcal{T} -paths, that is, deleting these vertices and edges each component of the remaining graph contains vertices from at most one $T_i \in \mathcal{T}$ ($i = 1, \dots, k$).

If $X, X_0 \subseteq V$ are disjoint, and $\{X_1, \dots, X_k\}$ is a partition of X , then $\mathcal{X} := (X; X_0, X_1, X_2, \dots, X_k)$ is said to be a \mathcal{T} -partition if $T_i \subseteq X_0 \cup X_i$, ($i = 1, \dots, k$). The *value* of this \mathcal{T} -partition \mathcal{X} is defined to be

$$\text{val}_{G, \mathcal{T}}(\mathcal{X}) := |X_0| + \sum_{C \in \mathcal{C}} \left\lfloor \frac{1}{2}|C \cap X| \right\rfloor , \quad (3)$$

where the elements of \mathcal{C} are the vertex-sets of the components of the graph $G - X_0 - \bigcup_{i=1}^k E[X_i]$.

Theorem 4.

$$\mu(G, \mathcal{T}) = \min \text{val}_{G, \mathcal{T}}(\mathcal{X}) , \quad (4)$$

where the minimum is taken over all \mathcal{T} -partitions \mathcal{X} .

It is easy to see that the two forms of Mader's theorem are equivalent: indeed, in the original variant Theorem 3, the set U_0 and the edges induced by the sets U_i ($i = 1, \dots, k$) block all the \mathcal{T} -paths, that is, all the components of the remaining graph contain terminal vertices in at most one of the T_i 's. It is straightforward now to define the sets of Theorem 4, and similarly, a dual solution of the latter is easy to define in terms of the former.

For a \mathcal{T} -partition $\mathcal{X} := (X; X_0, X_1, X_2, \dots, X_k)$, a component K of $G - X_0 - \bigcup_{i=1}^k E[X_i]$ is said to be *odd* (*even*) if $|K \cap X|$ is an odd (even) number. Set $K \cap X$ is denoted by $B(K)$ and is called the *border* of K . Denote $C(\mathcal{X})$ the union of the even components, $D(\mathcal{X})$ the union of the odd components. A \mathcal{T} -partition $\mathcal{X} := (X; X_0, X_1, X_2, \dots, X_k)$ is called a *Mader-partition* if $\text{val}_{G, \mathcal{T}}(\mathcal{X}) = \mu(G, \mathcal{T})$.

2 What Does an Optimal Partition Exclude?

In every minimax theorem the optimal solutions on the ‘max’ side (primal) and the optimal solutions on the ‘min’ side (dual) have a particular structure; this is a consequence of the trivial \leq inequality between these two, and if the equality holds, then a bunch of equalities are also implied (complementary slackness).

We first wish to exhibit (mainly with the two joined figures) the corresponding combinatorial structure, called ‘complementary slackness’ in linear programming, for Mader’s theorems. Moreover, like in the special cases exhibited in the Introduction for analogy, these conditions do imply some evident exclusions. The main goal of this section is to state these exclusions: like for directed accessibility or matchings the Edmonds-Gallai type structure will be the most exclusive Mader-set.

First consider the edge-disjoint case. Let \mathcal{P} be a family of edge-disjoint T -paths and $\mathcal{A} = \{A_1, \dots, A_k\}$ be a T -partition, and

$$|\mathcal{P}| = |A_0| + \sum_{C \in \mathcal{C}} \left\lfloor \frac{1}{2} d_G(C) \right\rfloor,$$

using the notations of Theorem 2.

It can be immediately checked from Mader’s theorem that $P \in \mathcal{P}$ touches only two different A_i -s, and either it goes directly from one to the other through an edge between the two, or it goes to an even or an odd component K of $G - \cup \mathcal{A}$ and has two common edges with the cut $\delta_G(X)$. (See Fig. 1.)

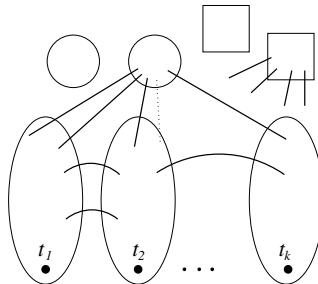


Fig. 1. An optimal T -partition

In a maximum family \mathcal{P} of pairwise edge-disjoint T -paths some vertices s might be the endpoints of an additional half T -path P , edge-disjoint from all the paths in the family. A vertex s will be called i -rooted ($t_i \in T$), if there is a family \mathcal{P} for which the endpoint of P in T is t_i .

Theorem 5 (Exclusion theorem for T -paths). *Let $\mathcal{A} := \{A_1, A_2, \dots, A_k\}$ be an optimal T -partition, A_0 the set of edges whose two endpoints are in A_i and A_j respectively, and $i \neq j$; $A := \bigcup_{i=1}^k A_i$, and the elements of \mathcal{C} are the vertex-sets of the components of the graph $G - A$.*

- (i) If $v \in C \in \mathcal{C}$, and $d_G(C)$ is even, then v is not i -rooted for any $i = 1, \dots, k$.
- (ii) If $v \in A_i$, then v is not j -rooted for $j \in \{1, \dots, k\}$, $j \neq i$.

Proof. Immediately follows from Mader's theorem (Theorem 2). \square

Next consider the vertex-disjoint case. It is useful to look at Theorem 4 for deducing the condition of equality, and then the corresponding exclusions.

Let \mathcal{P} be a maximum family of disjoint \mathcal{T} -paths and \mathcal{X} be a \mathcal{T} -partition for which equality holds in Theorem 4. Then $P \in \mathcal{P}$ either goes through vertices of exactly two different X_i, X_j ($1 \leq i < j \leq k$) and two vertices of the border of an even or odd component, or contains exactly one vertex of X_0 . If P has a vertex v in X_0 , then the segment of P between one end-node and v is divided into two parts. The first part (it maybe empty) consists of the segment of P which traverse an odd component K of \mathcal{X} traversing exactly one node of the border of K , then a part which is included in X_i ($i > 0$) and contains at most one vertex on a border of any odd components and no vertex on a border of any even components.

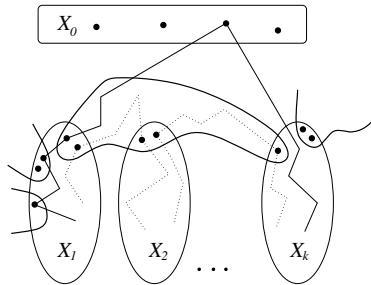


Fig. 2. A Mader-partition

Vertices s that are not covered by some maximum family of pairwise vertex-disjoint \mathcal{T} -paths might be the endpoints of an additional half \mathcal{T} -path P vertex-disjoint from all the paths in the family. Denote the endpoint of P in T by t , and let $t \in T_i$ ($i \in \{1, \dots, k\}$). Such a vertex s will be called i -rooted. If a vertex is not i -rooted for any $i \in \{1, \dots, k\}$, we will say it is 0-rooted. If a vertex is not i -rooted, but it is in T_i or has an i -rooted neighbor, we will say it is i -touched.

A vertex $t \in T_i$ is i -rooted if and only if it is not covered by a maximum family of vertex disjoint \mathcal{T} -paths. If it is covered by every maximum family, then by definition, it is i -touched.

Theorem 6 (Exclusion theorem for \mathcal{T} -paths). Let $\mathcal{X} := (X; X_0, X_1, X_2, \dots, X_k)$ be a Mader-partition, and \mathcal{C} the set of components with cardinality at least 2 of the graph $G - X_0 - \bigcup_{i=1}^k E[X_i]$.

- (i) If $v \in C \in \mathcal{C}$, where C is an even component, or $v \in X_0$, then v is not i -rooted for any $i = 1, \dots, k$.

- (ii) If $v \in X_i$ and is not in a border of an odd component, then for $j \in \{1, \dots, k\}$, $j \neq i$, v is neither j -rooted nor j -touched.
- (iii) If $v \in X_i$ and is in a border of an even component, then for $j \in \{1, \dots, k\}$, $j \neq i$, v is not j -touched.
- (iv) If $v \in X_i$ and is in a border of an odd component, then for $j \in \{1, \dots, k\}$, $j \neq i$, v is not j -rooted.

Proof. Immediately follows from Mader's theorem (Theorem 4). \square

3 Edge-Disjoint Paths

In this section we prove the structure theorem for maximum sets of edge-disjoint paths.

The terms and notations we introduce here are local in this section. (We will use the same or similar terminology in the rest of the paper, but in a different sense: in the definition of the terms 'edge-disjoint' will be replaced by 'vertex-disjoint'.)

The reader who aims at a quick direct understanding of the most general claims can skip this section: the results we are proving will be particular cases of the theorems on vertex-disjoint paths in the following section. However, it may be helpful to see the occurrence of the main ideas in a particular case.

Let $T = \{t_1, t_2, \dots, t_k\}$. The set of vertices that are i -rooted and not j -rooted if $j \neq i$ will be denoted by V_i . Let us denote the vertices that are both i -rooted and j -rooted for some $i \neq j$ by V_∞ . Vertices that are not i -rooted for any $i \in T$ are called 0-rooted. V_0 denotes the union of these vertices. Let \mathcal{V} denote the T -partition $\{V_1, \dots, V_k\}$.

By using Mader's Theorem 2, we will prove the following structure theorem.

Theorem 7 (Structure theorem of edge-disjoint paths). *The set \mathcal{V} is an optimal T -partition,*

V_∞ is the union of the odd components of \mathcal{V} and V_0 is the union of the even components of \mathcal{V} .

Proof. Let $\mathcal{X} := \{X_1, X_2, \dots, X_k\}$ be an optimal T -partition (i.e., whose value is μ^*) for which the cardinality of the union of the odd components is minimal and among these minimize the cardinality of the union of $\bigcup_{i=1}^k X_i$. By Mader's Theorem 2 such a T -partition exists.

We will prove that $\mathcal{V} = \mathcal{X}$. Let $D(\mathcal{X})$ ($C(\mathcal{X})$) denote the union of the odd (resp. even) components of \mathcal{X} .

$V - T$ is partitioned by the three sets $C(\mathcal{X}), \bigcup_{i=1}^k X_i - T, D(\mathcal{X})$, and on the other hand it is also partitioned by the three sets $V_0, \bigcup_{i=1}^k V_i, V_\infty$. We will prove the following three containment relations

- (1)* $C(\mathcal{X}) \subseteq V_0$,
- (2)* $X_i \subseteq V_i$ for $1 \leq i \leq k$,
- (3)* $D(\mathcal{X}) \subseteq V_\infty$.

It follows that each of the three classes of the first partition is contained in a corresponding class of the second partition, hence equalities follows throughout, that is, in (1)*, (2)* and (3)* as well.

(1)* follows by Mader's theorem immediately. (See Theorem 5 (i).)

Now we prove (2)*.

If there are edges between X_i and X_j , then subdivide them with single nodes, so we may assume that there are no edges between X_i and X_j for $1 \leq i < j \leq k$ (by Mader's theorem these vertices of degree 2 are always covered by a maximum set of edge-disjoint T -paths).

Now we define graph $G^* = (V^*, E^*)$. V^* is obtained by shrinking the components of $V - \bigcup_{i=1}^k X_i$ to single nodes and E^* is obtained by deleting the loops after the shrinkings.

Let us define the auxiliary graph $G' = (V', E')$ and capacities on the edges. Let $V' := V^* \cup \{r, s\}$, $E' := E^* \cup \{rt_i : 1 \leq i \leq k\} \cup \{ct : c \text{ is a node of } G^* - \bigcup_{i=1}^k X_i\}$. Let the capacity of the new edges incident to r be infinity, of an edge cs be $2 \cdot \left\lfloor \frac{d_{G^*}(c)}{2} \right\rfloor$, and of the other edges be one.

Let Y be the minimal (r, s) -cut in G' for which the cardinality of Y is minimum. Obviously, it has capacity $2\text{val}_{G', T}^*(\mathcal{X}) = 2\mu^*$.

Claim. $X_i \subseteq Y$ for all $1 \leq i \leq k$.

Proof. Since Y is a minimal minimum cut, it is clear that Y does not contain any node c which was obtained by shrinking an even component, furthermore if node $c \in Y$ and c was obtained by shrinking an odd component, then every edge $e \in E \cap E'$ incident to c has other endnode in $Y \cap \bigcup_{i=1}^k X_i$.

Suppose indirectly that $v \in X_i$ is not in Y . Then let $\mathcal{X}_Y := \{X_i \cap Y : 1 \leq i \leq k\}$. \mathcal{X}_Y is a T -partition and $|\bigcup_{X \in \mathcal{X}_Y} X|$ is strictly smaller than $|\bigcup_{X \in \mathcal{X}} X|$. Let l denote the number of nodes of G' in Y obtained by shrinking an odd component of \mathcal{X} . Let $q_{G^*}(\mathcal{X}_Y)$ denote the number of odd components of T -partition \mathcal{X}_Y in G^* .

Now we have

$$\begin{aligned} 2\mu^* &= d_{G'}(Y) = \sum_{i=1}^k d_{G'}(X_i \cap Y) - l \geq \\ &\sum_{i=1}^k d_{G^*}(X_i \cap Y) - q_{G^*}(\mathcal{X}_Y) = 2\text{val}_{G^*, T}^*(\mathcal{X}_Y) \geq 2\mu^*. \end{aligned}$$

Hence equality holds throughout, in particular, $\text{val}_{G^*, T}^*(\mathcal{X}_Y) = \mu^*$ and $l = q_{G^*}(\bigcup_{i=1}^k X_i)$, that is, after blowing up the shrunked components, T -partition \mathcal{X}_Y contradicts the choice of \mathcal{X} . \square

Proof of (2)*. Suppose indirectly that $u \in X_i$ is not in V_i . Let e denote a new edge us . Then the maximum value of an (r, s) -flow in G' does not change, hence there is a minimum cut Z which does not contain u . Since $Y \cap Z$ is also a minimum cut, it contradicts to Claim 3. \square

Proof of (3)*. We may assume that $X_i = \{t_i\}$ for all i and $G - T$ is a single odd component C .

Let v be a vertex in C . We may suppose that there is a node t_i such that after adding edge $e = vt_i$ to G , $\mu_{G+e,T}^* = \mu_{G,T}^* = \frac{d_G(C)-1}{2}$, that is, by Mader's Theorem 2, there is a T -partition \mathcal{F} such that $\text{val}_{G,T}(\mathcal{F}) = \frac{d_G(C)-1}{2} = \frac{d_{G+e}(C)-2}{2}$. Since the union of the odd components of \mathcal{F} is strictly smaller than $|C|$, it contradicts the choice of \mathcal{X} . \square

4 Vertex-Disjoint Paths

In this section we state and prove the structure theorem of vertex-disjoint paths. We first complete the (uniquely determined) classification of the vertices – started in Sect. 2 that leads to such a theorem.

In order to avoid a superfluous distinction of cases, it might be useful to note that the above distinction of touched terminal and non-terminal points is not really necessary: for every $t \in T$ introduce two new vertices, t_1, t_2 and join both to t and to nothing else. Replace $t \in T_i$ by $\{t_1, t_2\}$ for all $t \in T_i$, to get T'_i , and do this for all $i = 1, \dots, k$. Now every terminal point is i -rooted for some i (that is, it is in V^+ , which will be defined soon), and the status of the points in G did not change. In particular, if $t \in T_i$, then t certainly has the i -rooted neighbors t_1, t_2 , so it is at least i -touched, and it is i -rooted if and only if it was before, that is, if and only if t is not covered by every maximum path packing. (The gadget does not do anything else than realize physically what the definition provides anyway.)

Clearly, if $v \in V$ is i -rooted, then $\mu(G, T_1, \dots, T_j \cup \{v\}, \dots, T_k) = \mu(G, T) + 1$ for all $j \neq i$. If \mathcal{X} is an optimal T -partition, and $v \in X_i$, then the statement is reversible, since then v cannot be rooted in any other class but T_i .

Define the following sets of vertices.

- $C^* := \{v \in V : v \text{ is 0-rooted and is not } i\text{-touched for any } i\}$
- $D^* := \{v \in V : v \text{ is both } i\text{-rooted and } j\text{-rooted for some } i \neq j\}$
- $V_0 := \{v \in V : v \text{ is 0-rooted, and is both } i\text{-touched and } j\text{-touched for some } i \neq j\}$
- $V_i^* := \{v \in V : v \text{ is } i\text{-rooted and neither } j\text{-rooted, nor } j\text{-touched for any } j \neq i\}$
- $V_i^C := \{v \in V : v \text{ is 0-rooted, } i\text{-touched, and not } j\text{-touched for any } j \neq i\}$
- $V_i^D := \{v \in V : v \text{ is } i\text{-rooted, not } j\text{-rooted for all } j \neq i, \text{ and } j\text{-touched for some } j \neq i\}$
- $V_i := V_i^* \cup V_i^C \cup V_i^D$
- $V^+ := \bigcup_{i=1}^k V_i, C := C^* \cup \bigcup_{i=1}^k V_i^C, D := D^* \cup \bigcup_{i=1}^k V_i^D.$

It is easy to see that the sets above define a partition of V . We will see that $\mathcal{V} := \mathcal{V}(G, T) := \{V^+; V_0, V_1, \dots, V_k\}$ is a T -partition. Theorem 6 is the ‘trivial’ exclusion part of this.

Theorem 8 (Structure theorem of vertex-disjoint paths).

$\mathcal{V}(\mathcal{T}) = (V^+; V_0, V_1, V_2, \dots, V_k)$ is a \mathcal{T} -partition, $\mu(G, T) = \text{val}_{G, T}(\mathcal{V})$. Furthermore,

C is the union of the even components of $G - V_0 - \bigcup_{i=1}^k E[V_i]$, and

D is the union of the odd components of $G - V_0 - \bigcup_{i=1}^k E[V_i]$.

Proof. Let $\mathcal{X} := \{X; X_0, X_1, X_2, \dots, X_k\}$ be a \mathcal{T} -partition of V of value μ for which $X_0 \cup C(\mathcal{X})$ is inclusionwise maximal, and among these, maximize X (again, with respect to inclusion).

By Mader's Theorem 3 such a \mathcal{T} -partition \mathcal{X} exists. We will prove that $\mathcal{X} = \mathcal{V}(\mathcal{T})$. We keep the notation $\mathcal{V}(\mathcal{T}) = (V^+; V_0, V_1, \dots, V_k)$, $C = C(\mathcal{T})$, $D = D(\mathcal{T})$; in order to avoid confusion look again at the definition of these sets and realize that for the moment $C = C(\mathcal{T})$, $D = D(\mathcal{T})$ do not have anything to do with even or odd components; on the other hand $C(\mathcal{X})$ ($D(\mathcal{X})$) is the union of the even (odd) components of the graph $G - X_0 - \bigcup_{i=1}^k E[X_i]$.

We proceed by induction on $|V(G) \setminus T|$. If this is 0, then the theorem simplifies to the original Edmonds-Gallai structure theorem.

The proof will realize a project based on the following facts generalizing the edge-disjoint case: $V(G)$ is partitioned by the three sets $X_0 \cup C(\mathcal{X})$, $X \setminus C(\mathcal{X})$, and $D(\mathcal{X}) \setminus X$; on the other hand, $V(G)$ is also partitioned by the three sets $V_0 \cup C$, $V^+ \setminus C$, and D^* . We will prove the following three containment relations

- (1) $X_0 \cup C(\mathcal{X}) \subseteq V_0 \cup C$,
- (2) $X_i \setminus C(\mathcal{X}) \subseteq V_i \setminus V_i^C$ for all $i = 1, \dots, k$,
- (3) $D(\mathcal{X}) \setminus X \subseteq D^*$.

It follows that each of the three classes of the first partition is contained in a corresponding class of the second partition. The equalities follow throughout, that is, in (1), (2) and (3) as well. To prove the theorem we only have to prove $X_0 = V_0$ in addition to these equalities, and again, because of the equality in (1), this is equivalent to $X_0 \subseteq V_0$ and $C(\mathcal{X}) \subseteq C$. Let us prove these first, they are relatively simpler:

We prove $X_0 \subseteq V_0$, admitting (1), (2), (3). Let \mathcal{P} be an optimal path packing and $x \in X_0$. By complementary slackness (Fig. 2) x is contained in exactly one $P \in \mathcal{P}$, and P is vertex-disjoint from $C(\mathcal{X})$ and of $X_0 \setminus \{x\}$. Hence either one of the neighbors of x on P is in $D(\mathcal{X}) \setminus X$ – and by (3) in D^* –, or the two neighbors of x on P , denote them by a and b , are in $X_i \setminus C(\mathcal{X})$ and $X_j \setminus C(\mathcal{X})$ respectively, where $i \neq j$ (Fig. 2). Now by (2) a is then i -rooted, and b is j -rooted. Thus x is 0-rooted, i -touched and j -touched, that is, $x \in V_0$, as claimed.

We prove now $C(\mathcal{X}) \subseteq C$. Since $C(\mathcal{X}) \setminus X \subseteq C^*$ is obvious, all we have to prove is $C(\mathcal{X}) \cap X_i \subseteq V_i^C$ ($i = 1, \dots, k$). Fix $i \in \{1, \dots, k\}$, and let $v \in C(\mathcal{X}) \cap X_i$. By Theorem 6 (i) v is 0-rooted, and by (iii) it is neither j -rooted nor j -touched for $j \neq i$, proving the assertion.

Since $v \in X_0 \cup C(\mathcal{X})$ implies that v is 0-rooted, (1) is trivial.

We are remained with (2) and (3).

Let us prove now the most difficult part, (2). Let $v \in X_i \setminus C(\mathcal{X})$; then v is not j -rooted for any $j \neq i$ (since Theorem 6 (ii) and (iv)), whence the containment we want to prove is equivalent to the following:

Claim: Let $\mathcal{T}_v := (T_1, \dots, T_k, \{v\})$. Then $\mu(G, \mathcal{T}_v) = \mu(G, \mathcal{T}) + 1$.

Suppose for a contradiction $\mu(G, \mathcal{T}_v) = \mu(G, \mathcal{T})$. We know by induction that $\mathcal{V}(\mathcal{T}_v) =: (Y; Y_0, Y_1, \dots, Y_k, Y_v)$ is an optimal \mathcal{T}_v -partition. We show that

$$(4) \quad X_0 \cup C(\mathcal{X}) \subseteq Y_0 \cup C(\mathcal{T}_v) \cup U,$$

where U is the union of Y_v and of those components of $D(\mathcal{T}_v) = G - Y_0 - (E[Y_v] \cup \bigcup_{i=1}^k E[Y_i])$ that have a common vertex with Y_v .

The Claim will then be proved, since the optimal \mathcal{T} -partition \mathcal{Y} defined from $\mathcal{V}(\mathcal{T}_v)$ by deleting Y_v from the classes (and replacing Y by $Y \setminus Y_v$) has

$$X_0 \cup C(\mathcal{X}) \cup U \subseteq Y_0 \cup C(\mathcal{Y}).$$

Since $v \notin X_0 \cup C(\mathcal{X})$ and $v \in Y_0 \cup C(\mathcal{Y})$, \mathcal{Y} contradicts the maximal choice of $X_0 \cup C(\mathcal{X})$ if it is an optimal partition which we now prove.

If each optimal path packing in (G, \mathcal{T}_v) covers v , then $v \in Y_0 \cup C(\mathcal{T}_v)$ (and $U = \emptyset$). Since $\text{val}_{G, \mathcal{T}}(\mathcal{Y}) = \mu$, \mathcal{Y} contradicts the maximal choice of $X_0 \cup C(\mathcal{X})$.

If there is an optimal path packing in (G, \mathcal{T}_v) which does not cover v , then $v \in Y_v$. Hence every component K of $G - Y_0 - (E[Y_v] \cup \bigcup_{i=1}^k E[Y_i])$ having a nonempty intersection with Y_v is in $D(\mathcal{Y})$, and $|K \cap Y_v| = 1$. Hence $U \in C(\mathcal{Y})$, $\text{val}_{G, \mathcal{T}}(\mathcal{Y}) = \mu$, and \mathcal{Y} contradicts the maximal choice of $X_0 \cup C(\mathcal{X})$.

In order to prove the claim it is sufficient now to prove (4). We prove it by showing that the complement of the left hand side with respect to $V(G)$ contains the complement of the right hand side, that is, the following suffices:

Supposing that $u \notin U$ is j -rooted in (G, \mathcal{T}_v) for some $j \in \{1, \dots, k\}$, we show that it is also j -rooted in (G, \mathcal{T}) .

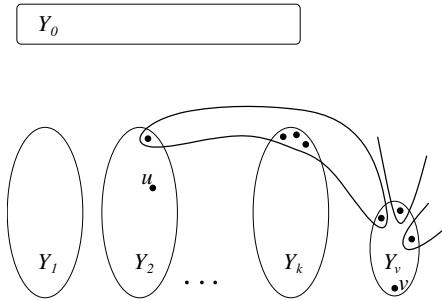


Fig. 3. Mader-partition if $Y_v \neq \emptyset$

Define $\mathcal{T}_{u,v} := (T_1, \dots, T_k, \{u, v\})$. We have $\mu(G, \mathcal{T}_{u,v}) = \mu(G, \mathcal{T}) + 1$, and if we have an optimal path packing that leaves either u or v uncovered, then we are done.

So u and v are covered by every maximal path packing in $(G, \mathcal{T}_{u,v})$. $\mathcal{V}(\mathcal{T}_v)$ shows that u (v) cannot be l -touched in $(G, \mathcal{T}_{u,v})$ for any $l \neq j$ ($l \neq i$). Hence by induction u and v are in $C(\mathcal{T}_{u,v})$. If u and v are not in the same even component of $C(\mathcal{T}_{u,v})$, then

$$\mu(G, \mathcal{T}) \leq \mu(G, \mathcal{T}_{u,v}) - 2,$$

which is impossible, therefore u , v are on the boundary of a component K of $C(\mathcal{T}_{u,v})$.

We know that K has an even number, say $2m$ boundary points including u and v ; in a system of $\mu(G, \mathcal{T}_{u,v}) = \mu(G, \mathcal{T}) + 1$ feasible $\mathcal{T}_{u,v}$ -paths these boundary points are matched by m paths, and in a system of $\mu(G, \mathcal{T})$ \mathcal{T} -paths the $2(m-1)$ boundary points are matched by $m-1$ paths (u and v are not terminal points any more, and therefore they are not boundary points).

By switching between the two sets of paths we get a bijection between maximum collections of \mathcal{T} -paths and maximum collections of $\mathcal{T}_{u,v}$ -paths. Both cover the same set of terminal vertices, except that the latter covers u , v , whereas the former does not.

It follows that $\mathcal{V}(\mathcal{T}_{u,v})$ is not only an optimal $\mathcal{T}_{u,v}$ -partition, but also an optimal \mathcal{T} -partition, and $\mathcal{V}(\mathcal{T}_{u,v}) = \mathcal{V}(\mathcal{T})$, contradicting the choice of \mathcal{X} .

(3) is now simpler to prove:

Indeed, let $v \in D(\mathcal{X}) \setminus X$, let K be the (odd) component of $D(\mathcal{X})$ containing v , and define the path packing problem (K, \mathcal{S}^i) as follows: $\mathcal{S}^i = (S_1, \dots, S_i \cup \{v\}, \dots, S_k)$, where $S_i = X_i \cap V(K)$ ($i = 1, \dots, k$). Denote $2m+2$ the number of terminal points in \mathcal{S}^i ; since \mathcal{X} is an optimal \mathcal{T} -partition, $\mu(K, \mathcal{S}) = m$ or $m+1$.

If for some $i = 1, \dots, k$, $\mu(K, \mathcal{S}^i) = m$, then there exists in K an \mathcal{S}^i -partition \mathcal{Z} of value m . Since the \mathcal{S}^i -partition $\mathcal{Y} = (\{v\} \cup \cup_{i=1}^k S_i; Y_0 = \emptyset, \mathcal{S}^i)$ has value $m+1$, in \mathcal{Z} either $Z_0 \neq \emptyset$ or there exists $i \in \{1, \dots, k\}$ such that $|Z_i| > |S_i|$.

Combining \mathcal{Z} with \mathcal{X} in the obvious way, we get in the former case a \mathcal{T} -partition where $X_0 \cup C(\mathcal{X})$ increases, whereas in the latter case a \mathcal{T} -partition where X increases, contradicting in both cases the choice of \mathcal{X} .

Therefore, for all $i = 1, \dots, k$, $\mu(K, \mathcal{S}^i) = m+1$. For a fixed i , $\mu(K, \mathcal{S}^i) = m+1$ means exactly that there exists $j \neq i$ so that v is j -rooted in K , and this j is not unique, since $\mu(K, \mathcal{S}^j) = m+1$ also holds. Since according to the Claim every $s \in S_i$ is i -rooted, we can combine the maximum path packing and the half path proving that s is i -rooted with the appropriate path packings in (K, \mathcal{S}^i) showing that v is j -rooted in (G, \mathcal{T}) . Using that v is also l -rooted for some $l \neq j$ in K , we get a path packing proving that v is l -rooted in (G, \mathcal{T}) .

Now the claim and the theorem is proved. \square

5 Generalizations

Suppose we are given a function $c : V(G) \rightarrow \mathbb{N}$, and we want to maximize the number of \mathcal{T} -paths so that vertex $v \in V(G)$ is contained in at most $c(v)$ paths. Let us denote by $\nu(G, T, c)$ the maximum.

Theorem 9.

$$\nu(G, T, c) = \min c(V_0) + \sum_{C \in \mathcal{C}} \left\lfloor \frac{c(\bigcup_{i=1}^k V_i \cap C)}{2} \right\rfloor,$$

where $V_0 \subseteq V(G)$, V_i is a subpartition of $V(G) \setminus V_0$, $V_0 \cup V_i \supseteq T_i$, and \mathcal{C} is the collection of the components of $G - V_0 - \bigcup_{i=1}^k E[V_i]$.

Proof. Replicate each vertex – vertex v $c(v)$ times. (All copies are adjacent to the same vertex-set, and non-adjacent among them.) Note that all the copies of a vertex are in the same class of the structure theorem, and the theorem follows. The statement is also easy directly from Mader's Theorem 4. \square

In other words, putting weights on vertices does not generalize the problem, but the statement can be useful in algorithmic considerations.

In the following we formulate a common generalization of Mader's T -path problem and Cunningham and Geelen's path-matching theorem [3]. We call this problem the T -path-matching problem.

Let $G = (V, E)$ be a graph and T_1, \dots, T_k disjoint subsets of V , denote $T := \{T_1, \dots, T_k\}$, $T := \bigcup_{i=1}^k T_i$. A T -path-matching is a union of edges (called matching edges) and T -paths and paths with one end in T and the other not in T , all vertex-disjoint. The value of a path-matching is the number of its edges plus the number of its matching edges, that is, the matching edges count twice. The perfect version of this problem was introduced by Z. Szigeti [20].

A T -partition is a family of pairwise disjoint sets $X_0, X_1, \dots, X_k \subseteq V(G)$ so that $X_0 \cup X_i \supseteq T_i$. (It is not necessarily a partition of $V(G)$.) The components of the graph $(G - X_0) - \bigcup_{i=1}^k E[X_i]$ will be called the *components* of this T -partition. Let us denote by \mathcal{C} the family of these components.

Let $X := \bigcup_{i=1}^k X_i$.

We denote by ω the number of odd components that do not meet X .

For a component C which meets X , $\omega_i(C)$ denotes the number of components of $C - X$ of odd cardinality having neighbors only in $X_i \cup X_0$ for $i = 1, 2, \dots, k$. Let $\omega(C) := \sum_{i=1}^k \omega_i(C)$.

The value of a T -partition is defined to be

$$\text{val}_{G, T}(\mathcal{X}) := |V \setminus T| + (|X_0| - \omega) + \sum_{C \in \mathcal{C}} \left\lfloor \frac{|C \cap X| - \omega(C)}{2} \right\rfloor.$$

Conjecture 1. The maximum value of a T -path-matching is equal to

$$\min \text{val}_{G, T}(\mathcal{X}),$$

where the minimum is taken over all T -partitions \mathcal{X} .

The proof of the so called ‘trivial’ part (which is indeed, easy, but requires more concentration than for simpler min-max theorems) can be found in [18].

Isn’t it challenging to search for a structure algorithm for T -path-matchings (in polynomial time)? This conjecture can be reduced to matchings in the particular case when every class of T has at most one element, which motivates an adaptation of Schrijver’s approach [14].

Acknowledgement. We thank Zoltán Szigeti for his valuable comments.

References

1. Cornuéjols, G., Hartvigsen, D., Pulleyblank, D.: Packing subgraphs in a graph. Oper. Res. Letter, **1**/4 (1981/82) 139–143
2. Cunningham, W.H.: Matching, matroids and extensions. Math. Program. Ser. B **91**/3 (2002) 515–542
3. Cunningham, W.H., Geelen, J.F.: The optimal path-matching problem. Combinatorica **17**/3 (1997) 315–336
4. Edmonds, J.R.: Maximum matching and a polyhedron with 0,1-vertices. J. Res. Nat. Bur. Standards Sect. B (1968) 125–130
5. Edmonds, J.R., Johnson, E.L.: Matching: a well-solved class of integer linear programs. in Combinatorial Structures and Their Applications, Calgary, Alberta, (1969), Guy, Hanani, Sauer, Schönheim eds.
6. Frank, A., Szegő, L.: A Note on the Path-Matching Formula. J. of Graph Theory **41**/2 (2002) 110–119
7. Gallai, T.: Neuer Beweis eines Tutte’schen Satzes. Magyar Tud. Akad. Mat. Kutató Int. Közl., **8** (1963) 135–139
8. Gallai, T.: Maximale Systeme unabhängiger Kanten. Magyar Tud. Akad. Mat. Kutató Int. Közl., **9** (1965) 401–413
9. Lovász, L.: On the structure of factorizable graphs. Acta Math. Acad. Sci. Hungar. **23** (1972) 179–195
10. Lovász, L.: The factorization of graphs II. Acta Math. Acad. Sci. Hungar. **23** (1972) 223–246
11. Lovász, L., Plummer, M.D.: Matching Theory. Akadémiai Kiadó, Budapest, 1986.
12. Mader, W.: Über die Maximalzahl kantendisjunkter A -Wege. Arch. Math. (Basel) **30**/3 (1978) 325–336
13. Mader, W.: Über die Maximalzahl kreuzungsfreier H -Wege. Archiv der Mathematik (Basel) **31** (1978) 387–402
14. Schrijver, A.: A short proof of Mader’s S -paths theorem, Journal of Combinatorial Theory Ser. B **82** (2001) 319–321
15. Sebő, A.: Finding the T -join structure of graphs. Mathematical Programming, **36** (1986) 123–134
16. Sebő, A.: Factors of Graphs: Structures and Algorithms. Candidate’s thesis, Hungarian Academy of Sciences, Budapest 1987.
17. Sebő, A.: Undirected distances and the postman structure of graphs. Journal of Combinatorial Theory Ser. B **49** (1990) 10–39
18. Sebő, A., Szegő, L.: The path-packing structure of graphs. Egres Technical Report (2003), <http://www.cs.elte.hu/egres/tr/egres-03-07.ps>
19. Spille, B., Szegő, L.: A Gallai-Edmonds-type Structure Theorem for Path-Matchings. J. of Graph Theory (2002), to appear
20. Szigeti, Z., personal communication (1999)

More on a Binary-Encoded Coloring Formulation

Jon Lee¹ and François Margot²

¹ IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A.

jonlee@us.ibm.com

² Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA

fmargot@andrew.cmu.edu

Abstract. We further develop the 0/1 ILP formulation of Lee for edge coloring where colors are encoded in binary. With respect to that formulation, our main contributions are: (i) an efficient separation algorithm for general block inequalities, (ii) an efficient LP-based separation algorithm for stars (i.e., the all-different polytope), (iii) introduction of matching inequalities, and (iv) introduction of switched path inequalities and their efficient separation, (v) a complete description for paths, (vi) promising computational results.

Introduction

Let G be a simple finite graph with vertex set $V(G)$ and edge set $E(G)$ and let $m := |E(G)|$. For $v \in V(G)$, let $\delta(v) := \{e \in E(G) : e \text{ is incident to } v\}$. Let $\Delta(G) := \max\{|\delta(v)| : v \in V(G)\}$. Let c be a positive integer, and let $C := \{0, 1, \dots, c - 1\}$.

A *proper edge c -coloring* of G is a function Φ from $E(G)$ to C , so that Φ restricted to $\delta(v)$ is an injection, for all $v \in V(G)$. Certainly, a proper edge c -coloring can not exist if $c < \Delta(G)$. Vizing [11] proved that a proper edge c -coloring always exists when $c > \Delta(G)$. Holyer [6] proved that it is NP-Complete to decide whether G has a proper edge $\Delta(G)$ -coloring (even when $\Delta(G) = 3$).

Lee [7] developed a 0/1 integer linear programming formulation of the feasibility problem of determining whether G has a proper edge c -coloring based on the following variables: For each edge $e \in E(G)$, we use a string of n 0/1-variables to encode the color of that edge (i.e., the n -string is interpreted as the binary encoding of an element of C). Henceforth, we make no distinction between a color (i.e., an element of C) and its binary representation in $\{0, 1\}^N$.

Let $N := \{0, \dots, n - 1\}$. For $X \in \mathbb{R}^{E(G) \times N}$, we let x_e denote the row of X indexed by e and x_e^i denote the entry of X in row e and column i ($e \in E(G)$, $i \in N$). We define the *n -bit edge coloring polytope* of G as

$$Q_n(G) := \text{conv}\left\{X \in \{0, 1\}^{E(G) \times N} : x_e \neq x_f, \forall \text{ distinct } e, f \in \delta(v), \forall v \in V(G)\right\}.$$

The graph G is a *star* if there is a $v \in V(G)$ such that $E(G) = \delta(v)$. If G is a star, then we call $Q(m, n) := Q_n(G)$ the *all-different polytope* (as defined in [7]). In this case, we let $M := E(G)$. For a general graph G , the all-different polytope

is the fundamental “local” modeling object for encoding the constraint that Φ restricted to $\delta(v)$ is an injection, for all $v \in V(G)$. Note that this type of constraint is present in several combinatorial problems besides edge coloring: vertex coloring, timetabling, and some scheduling problems for example. Although the focus of this paper is on edge coloring, the results of Sections 1 and 2 are relevant in all such situations.

For determining whether G has a proper edge c -coloring, we choose $n := \lceil \log_2 c \rceil$, so we are using only $\sim m \log c$ variables, while the more obvious assignment-based formulation would require mc variables. A rudimentary method for allowing only c of the possible 2^n colors encoded by n bits is given in [7]; A much more sophisticated method for addressing the case where the number of colors c is not a power of two (i.e., $2^{n-1} < c < 2^n$) can be found in [4].

One difficulty with this binary-encoded model is to effectively express the all-different constraint at each vertex — that is, to give a computationally-effective description of the all-different polytope by linear inequalities. In Sections 1 and 2, we describe progress in this direction. In Sections 3 and 4, we describe progress for general graphs (i.e., not just stars). In Section 5, we describe our implementation and results of computational experiments.

In the remainder of this section, we set some notation and make some basic definitions. For $x_e \in \mathbb{R}^N$ with $\mathbf{0} \leq x_e \leq \mathbf{1}$, and $S, S' \subseteq N$ with $S \cap S' = \emptyset$, we define the *value of* (S, S') on e as

$$\mathbf{x}_e(S, S') := \sum_{i \in S} x_e^i + \sum_{i \in S'} (1 - x_e^i).$$

When $S \cup S' = N$, the ordered pair (S, S') is a partition of N . A *t-light partition* for $x_e \in \mathbb{R}^N$ is a partition (S, S') with $\mathbf{x}_e(S, S') < t$. An *active partition* for $e \in E(G)$ is a 1-light partition. For $E' \subseteq E(G)$, we define the *value of* (S, S') on E' as

$$\mathbf{x}_{E'}(S, S') := \sum_{e \in E'} \mathbf{x}_e(S, S').$$

1 Separation for General Block Inequalities

For $1 \leq p \leq 2^n$, p can be written uniquely as

$$p = h + \sum_{k=0}^t \binom{n}{k}, \text{ with } 0 \leq h < \binom{n}{t+1}.$$

The number t (resp., h) is the *n-binomial size* (resp., *remainder*) of p . Then let

$$\kappa(p, n) := (t+1)h + \sum_{k=0}^t k \binom{n}{k}.$$

Let S, S' be a partition of N and let L be a subset of M . Then $X \in Q(m, n)$, must satisfy the *general block inequalities* (see [7]):

$$(GBI) \quad \kappa(|L|, n) \leq \mathbf{x}_L(S, S').$$

In fact, general block inequalities are facet-describing for the all-different polytope when the n -binomial remainder of $|L|$ is not zero [7].

Lemma 1. *Let p satisfy $1 \leq p \leq 2^n$, and let t be the n -binomial size of p . Then for $x_e \in \mathbb{R}^N$ with $\mathbf{0} \leq x_e \leq \mathbf{1}$, at most $4(n+1)^2p^2 + (n+1)p$ partitions are $(t+1)$ -light for x_e .*

Proof. Let (S_1, S'_1) and (S_2, S'_2) be two $(t+1)$ -light partitions with $d := |S_1 \Delta S_2|$ maximum. Without loss of generality, we can assume that $S_1 = \emptyset$ by replacing x_e^i by $1 - x_e^i$ for all $i \in S_1$. As

$$2(t+1) > \mathbf{x}_e(S_1, S'_1) + \mathbf{x}_e(S_2, S'_2) \geq d,$$

we have that $d \leq 2t+1$. The number T of possible $(t+1)$ -light partitions satisfies

$$T \leq \sum_{k=0}^{2t+1} \binom{n}{k}.$$

Using that, for all $k \leq n/2$, $\binom{n}{2k} \leq \binom{n}{k}^2$ and $\binom{n}{2k-1} \leq \binom{n}{k}^2$ and that, for nonnegative numbers a, b , we have $a^2 + b^2 \leq (a+b)^2$, we get

$$T \leq \left(2 \sum_{k=0}^{t+1} \binom{n}{k} \right)^2 + \sum_{k=0}^{t+1} \binom{n}{k}$$

By hypothesis, we have $\binom{n}{t+1} \leq n \binom{n}{t} \leq np$, and thus

$$\sum_{k=0}^{t+1} \binom{n}{k} \leq (p-h) + np \leq (n+1)p.$$

The result follows. \square

Note that computing all of the $(t+1)$ -light partitions for x_e in the situation of Lemma 1 can be done in time polynomial in p and n using Reverse Search [1]: The number of partitions in the output is polynomial in p and n , and Reverse Search requires a number of operations polynomial in p and n for each partition in the output.

We are led then to the following

SEPARATION ALGORITHM FOR GBI

- (0) Let $X \in [0, 1]^{M \times N}$, and let t be the n -binomial size of m .
- (1) For each $e \in M$, compute the set T_e of all $(t+1)$ -light partitions for x_e .
- (2) Then, for each partition (S, S') in $\cup_{e \in M} T_e$:
 - (2.a) Compute $F \subseteq M$ such that, for each $e \in F$, (S, S') is a $(t+1)$ -light partition for x_e .
 - (2.b) Order $F = \{e_1, \dots, e_f\}$ such that $\mathbf{x}_{e_i}(S, S') \leq \mathbf{x}_{e_{i+1}}(S, S')$ for $i = 1, \dots, f-1$.

- (2.c) If one of the partial sums $\sum_{i=1}^k \mathbf{x}_{\mathbf{e}_i}(S, S')$, for $k = 2, \dots, f$ is smaller than $\kappa(k, n)$, then $L := \{e_1, \dots, e_k\}$ and (S, S') generate a violated GBI for X .

By Lemma 1, it is easy to see that the algorithm is polynomial in p and n . We note that a much simpler algorithm can be implemented with complexity polynomial in p and 2^n : Replace $\cup_{e \in E'} T_e$ by the set of all 2^n possible partitions.

Theorem 1. *Let $X \in [0, 1]^{M \times N}$. If the algorithm fails to return a violated GBI for X , then none exists.*

Proof. Suppose that $L \subseteq M$ generates a violated GBI for partition (S, S') . Let s be the n -binomial size of $|L|$. Observe that $\kappa(|L|, n) - \kappa(|L|-1, n) \leq s+1$. We may assume that no proper subset of L generates a GBI for partition (S, S') . Then $\mathbf{x}_e(S, S') < s+1$ for all $e \in L$. As $s \leq t$, this implies that (S, S') is a $(t+1)$ -light partition for x_e , and the algorithm will find a violated GBI. \square

We can sharpen Lemma 1 for the case of $t = 0$ to obtain the following result.

Lemma 2. *Let $e \in M$ and $x_e \in \mathbb{R}^N$ with $\mathbf{0} \leq x_e \leq \mathbf{1}$. Then there are at most two active partitions for e . Moreover, if two active partitions, say (S_1, S'_1) and (S_2, S'_2) exist, then $|S_1 \Delta S_2| = 1$.*

Proof. $2 > \mathbf{x}_e(S_1, S'_1) + \mathbf{x}_e(S_2, S'_2) \geq |S_1 \Delta S_2|$ implies that $|S_1 \Delta S_2| = 1$. Moreover, we can not have more than two subsets of N so that the symmetric difference of each pair contains just one element. \square

Motivated by Lemma 2, we devised the following heuristic as a simple alternative to the exact algorithm.

SEPARATION HEURISTIC FOR GBI

- (0) With respect to x_e , compute its (at most) two active partitions and their values.
- (1) Then, for each partition (S, S') of N :
 - (1.a) Compute the set T of elements in M that have (S, S') as an active partition.
 - (1.b) Sort the $e \in T$ according to nondecreasing $\mathbf{x}_e(S, S')$, yielding the ordering $T = \{e_1, \dots, e_t\}$.
 - (1.c) If one of the partial sums $\sum_{i=1}^k \mathbf{x}_{\mathbf{e}_i}(S, S')$, for $k = 2, \dots, t$, is smaller than $\kappa(k, n)$, then $L := \{e_1, \dots, e_k\}$ and (S, S') generate a violated GBI for X .

The complexity is $O(2^n m \log m)$. Note that the heuristic is an exact algorithm if the n -binomial size of m is zero.

2 Exact LP-based Separation

In this section we describe an exact separation algorithm for the all-different polytope $Q(m, n)$. The algorithm is polynomial in m and 2^n . In many situations (e.g., edge coloring), we consider 2^n to be polynomial in the problem parameters (e.g., $\Delta(G)$); so the algorithm that we describe may be considered to be efficient in such situations.

Theorem 2. *Let \bar{X} be a point in $[0, 1]^{M \times N}$. There is an efficient algorithm that checks whether \bar{X} is in $Q(m, n)$, and if not, determines a hyperplane separating \bar{X} from $Q(m, n)$.*

Proof. The point \bar{X} is not in $Q(m, n)$ if and only if there is a solution to:

$$(I.1) \quad \sum_{i \in M} \sum_{j \in N} \pi_i^j \bar{x}_i^j > \sigma ;$$

$$(I.2) \quad \sum_{i \in M} \sum_{j \in N} \pi_i^j x_i^j \leq \sigma , \quad \forall x \in Q_I(m, n),$$

where $Q_I(m, n) := Q(m, n) \cap \{0, 1\}^{M \times N}$, $\pi \in \mathbb{R}^{M \times N}$, and $\sigma \in \mathbb{R}$.

We apply the ellipsoid method (see [5]) to (I.1-2). This results in a polynomial-time algorithm provided that we can solve the separation problem for (I.2) in polynomial time. We can write this latter separation problem for $(\bar{\pi}, \bar{\sigma})$ as

$$(\mathcal{P}) \quad z := \max \left\{ \sum_{i \in M} \sum_{j \in N} \bar{\pi}_i^j x_i^j : X \in Q_I(m, n) \right\},$$

where $z > \bar{\sigma}$ if and only if there exists a violated inequality of the type (I.2).

But (\mathcal{P}) is equivalent to

$$\begin{aligned} (\mathcal{P}') \quad & \max \sum_{i \in M} \sum_{k \in O} w_{ik} z_{ik} \\ & \text{s.t. } \sum_{i \in M} z_{ik} \leq 1 , \quad \forall k \in O , \\ & \sum_{k \in O} z_{ik} = 1 , \quad \forall i \in M , \\ & z_{ik} \in \{0, 1\} , \quad \forall i \in M, \forall k \in O , \end{aligned}$$

where

$$\begin{aligned} O &:= \{0, 1, \dots, 2^n - 1\} , \\ w_{ik} &:= \sum_{j \in N} \bar{\pi}_i^j \cdot \text{bit}_j[k] , \end{aligned}$$

$\text{bit}_j[k] :=$ the j th bit in the binary encoding of k .

The result follows since (\mathcal{P}') is just an ordinary assignment problem.

For computational purposes, we do not want to rely on the separation algorithm that is implied by the preceding proof, because it makes use of the ellipsoid method. We call a valid inequality $\sum_i \sum_j \pi_i^j x_i^j \leq \sigma$ *normalized* if $\sigma \leq 1$ and $-\mathbf{1} \leq \pi \leq \mathbf{1}$. Clearly, if a valid inequality separating \bar{X} from $Q(m, n)$ exists, then a normalized such inequality exists as well. In the complete version of this paper, we will describe a practical approach which is also theoretically efficient, using ideas similar to those of [9]. This approach yields an algorithm for producing maximally violated normalized cuts if any such cut exists. In Section 5, we refer to cuts produced in this way as *LP cuts* (LPC).

3 Matching Inequalities

Let S, S' be subsets of N with $S \cap S' = \emptyset$. The *optimal colors for* (S, S') are the colors $\mathbf{x} \in \{0, 1\}^N$ that yield $\mathbf{x}(S, S') = 0$. The set of optimal colors for (S, S') is denoted by $\mathcal{B}(S, S')$. Note that if (S, S') is a partition of N , then there is a unique optimal color which is the characteristic vector of S' . In general, if $|N \setminus (S \cup S')| = k$, then the set of optimal colors for (S, S') has 2^k elements (it is the set of vertices of a k -dimensional face of $[0, 1]^N$). Note that if $\mathbf{x} \in \{0, 1\}^N$ is a not an optimal color for (S, S') , then $\mathbf{x}(S, S') \geq 1$.

Proposition 1. *Let $E' \subseteq E(G)$, and let $F \subseteq E'$ be a maximum matching in the graph induced by E' . Let (S, S') be a partition of N . The matching inequality (induced by E')*

$$(MI) \quad \mathbf{x}_{E'}(S, S') \geq |E' \setminus F|$$

is valid for $Q_n(G)$.

Proof. At most $|F|$ edges in E' can have the optimal color for (S, S') , and every other edge has a color contributing at least one to the left-hand side. \square

When E' is an odd cycle, the matching inequalities reduce to the so-called “type-I odd-cycle inequalities” (see [7] which introduced these latter inequalities and [8] which provided an efficient separation algorithm for them).

A MI is *dominated* if it is implied by MI on 2-connected non-bipartite subgraphs and by GBI. The following proposition shows that it is enough to generate the non-dominated MI, provided that the GBI generated by the separation heuristic for GBI of Section 1 are all satisfied.

Proposition 2. *Let G' be the graph induced by E' . The MI induced by E' is dominated in the following cases:*

- (i) G' is not connected;
- (ii) G' has a vertex v saturated by every maximum matching in G' ;
- (iii) G' has a cut vertex v ;
- (iv) G' is bipartite.

Proof. (i) The MI is implied by those induced by the components of G' .

(ii) The MI is implied by the MI on $G' - v$ and the GBI for $\delta(v) \cap E'$.

(iii) Let G_1 and G_2 be a partition of E' sharing only vertex v . By (ii), we can assume that there exists a maximum matching F of G with v not saturated by F . Then $E(F) \cap E(G_i)$ is a maximum matching in G_i for $i = 1, 2$. The MI is thus implied by the MI on G_1 and G_2 .

(iv) By the König's theorem, the cardinality of a minimum vertex cover of G' is equal to the cardinality k of a maximum matching F of G' . It is then possible to partition the edges of G' into k stars, such that star i has k_i edges. If the GBI inequalities for the stars are all satisfied, then summing them up yields:

$$\mathbf{x}_{\mathbf{E}(G')}(S, S') \geq \sum_{i=1}^k (k_i - 1) = |E(G')| - k = |E(G') \setminus M|,$$

and the MI induced by E' is also satisfied. \square

Recall that a *block* of a graph is a maximal 2-connected subgraph. Proposition 2 is the justification of the following:

SEPARATION HEURISTIC FOR MI

(0) Let \bar{X} be a point in $[0, 1]^{E(G) \times N}$.

(1) For each partition (S, S') of N :

(1.a) Compute the edges T for which (S, S') is an active partition.

(1.b) For each non-bipartite block of the graph G' induced by T :

(1.b.i) Compute a maximum matching $F(G')$ in G' .

(1.b.ii) Check if $\mathbf{x}_{\mathbf{E}(G')}(S, S') \geq |E(G') \setminus F(G')|$ is a violated matching inequality.

Complexity: Since each edge of G has at most two active partitions, all computations of active partitions take $O(nm)$ and all computations of non-bipartite blocks take $O(m)$. For one partition (S, S') , computing the maximum matchings takes $O(\sqrt{|V(G)|} m)$ [10]. The overall complexity is thus $O(2^n \sqrt{|V(G)|} m)$.

Note that ignoring edges e for which (S, S') is not an active partition does not prevent generation of violated matching inequalities: Suppose that e appears in a violated matching inequality $\bar{\mathbf{x}}_{\mathbf{E}(G')}(S, S') < |E(G') \setminus F(G')|$. Then $\bar{\mathbf{x}}_{\mathbf{E}(G')-e}(S, S') < |(E(G') - e) \setminus F(G' - e)|$ is also violated, as the left-hand side has been reduced by more than 1, while the right-hand side has been reduced by at most 1. The algorithm is nevertheless not exact, as we should generate MI for all 2-connected subgraphs, not only for blocks. In practice, the blocks are very sparse and rarely contain more than a few odd cycles. Enumerating the 2-connected non-bipartite subgraphs might thus be feasible.

4 Switched Walk Inequalities

Let $S, S' \subseteq N$ such that $S \cap S' = \emptyset$ and $|S \cup S'| \geq n - 1$. Then (S, S') is a *subpartition* of N .

Let (S_1, S'_1) be a subpartition of N . Let (S_2, S'_2) be a subpartition obtained from (S_1, S'_1) by performing the following two steps:

- (1) adding the only element not in $S_1 \cup S'_1$ (if any) either to S_1 or to S'_1 ; call the resulting partition (P_2, P'_2) .
- (2) removing at most one element from P_2 or at most one element from P'_2 .

Then (S_2, S'_2) is a *switch* of (S_1, S'_1) . Observe that $|\mathcal{B}(S_1, S'_1)| \leq 2$, that $|\mathcal{B}(S_2, S'_2)| \leq 2$ and that $|\mathcal{B}(S_1, S'_1) \cap \mathcal{B}(S_2, S'_2)| \geq 1$.

Let (e_1, \dots, e_k) be the ordered set of edges of a walk in G with $k \geq 2$. For $i = 1, \dots, k$, let (S_i, S'_i) be subpartitions of N such that

- (a) $|S_i \cup S'_i| = \begin{cases} n, & \text{if } i = 1, \text{ or } i = k; \\ n - 1, & \text{otherwise.} \end{cases}$
- (b) For $i = 1, \dots, k - 1$, (S_{i+1}, S'_{i+1}) is a switch of (S_i, S'_i) .
- (c) For all $j \in S_t$, if $[t+1, t']$ is a maximal interval such that for all $t+1 \leq i \leq t'$ we have $N - (S_i \cup S'_i) = \{j\}$, then $j \in S'_{t'+1}$ if and only if $t' - t$ is odd.
- (d) For all $j \in S'_t$, if $[t+1, t']$ is a maximal interval such that for all $t+1 \leq i \leq t'$ we have $N - (S_i \cup S'_i) = \{j\}$, then $j \in S'_{t'+1}$ if and only if $t' - t$ is even.

Then the walk and the set of subpartitions $(S_1, S'_1), \dots, (S_k, S'_k)$ form a *switched walk*.

Given a switched walk, the inequality

$$(SWI) \quad \sum_{i=1}^k \mathbf{x}_{\mathbf{e}_i}(S_i, S'_i) \geq 1$$

is a *switched walk inequality*.

Example 1. Let $N := \{0, 1, 2\}$. Consider the path of edges $(e_1, e_2, e_3, e_4, e_5)$. Associated with the sequence of edges of the path is the switched walk: $(\{0\}, \{1, 2\}), (\{0\}, \{2\}), (\{1\}, \{2\}), (\{1\}, \{0\}), (\{1, 2\}, \{0\})$. The given switched walk gives rise to the SWI:

$$\begin{array}{rccccc} +x_1^0 & +(1-x_1^1) & +(1-x_1^2) & & & \\ +x_2^0 & & & +(1-x_2^2) & & \\ & & +x_3^1 & & +(1-x_3^2) & \\ +(1-x_4^0) & & +x_4^1 & & & \\ +(1-x_5^0) & & +x_5^1 & +x_5^2 & & \geq 1 \end{array}$$

The only possibility for a 0/1 solution to violate this is to have $\mathbf{x}_{\mathbf{e}_i} = 0$ for $i = 1, 2, \dots, 5$. This implies that the color of e_1 must be 011. Then, of the two possible colors for e_2 (achieving $\mathbf{x}_{\mathbf{e}_2} = 0$), the only one that is different from the color of e_1 is 001. Similarly, e_3 must get color 101 and e_4 gets 100. But this is not different from the only color giving $\mathbf{x}_{\mathbf{e}_5} = 0$.

Theorem 3. *The SWI are valid for $Q_n(G)$.*

Proof. By induction on k . If $k = 2$, the SWI is a GBI and thus is valid.

Assume that there exists a SWI with $k \geq 3$ that is not valid for the edge coloring polytope. Then there exists a coloring of the edges in the walk W such that each edge $e_i \in W$ receives a color in $\mathcal{B}(S_i, S'_i)$. As subpartitions (S_i, S'_i) and (S_{i+1}, S'_{i+1}) are switches of each other, $|\mathcal{B}(S_i, S'_i) \cap \mathcal{B}(S_{i+1}, S'_{i+1})| \geq 1$, implying that once the color of e_i is chosen, then a unique color for e_{i+1} is available. As $\mathcal{B}(S_1, S'_1)$ contains a unique color, the coloring of edges in W is unique. It is then possible to transform each subpartition (S_i, S'_i) for $i = 2, \dots, k-1$ into a partition (T_i, T'_i) of N such that $\mathcal{B}(T_i, T'_i)$ is the color assigned to e_i .

We now prove by induction that $(T_{k-1}, T'_{k-1}) = (S_k, S'_k)$, yielding a contradiction. If $k = 3$, then if j was the element missing in (S_2, S'_2) then conditions c) and d) above imply that j is in S_1 if and only if j is in S'_3 . Since j is in S_1 if and only if j is in T'_2 , we get $T'_2 = S'_3$ and we are done.

If $k > 3$, then the same reasoning for the switched walk starting with edge e_2 and partition (T_2, T'_2) and ending with e_k with partition (S_k, S'_k) yields the result by induction. \square

Next, we state a result indicating the importance of the switched walk inequalities. The proof will appear in the complete version of this paper.

Theorem 4. *If P is a path and $n \geq 2$, then $Q_n(P)$ is described by the SWI and the simple bound inequalities $\mathbf{0} \leq X \leq \mathbf{1}$.*

We separate the SWI by solving m shortest path problems on a directed graph with at most $8(n+1)m$ nodes and $32n(n+1)^2m$ edges. The overall complexity of the separation is $O(mn^3 \log(mn))$.

5 Computational Results

We report preliminary results for Branch-and-Cut (B&C) algorithms using the GBI, LPC, MI and SWI. The code is based on the open source BCP code with the open-source LP solver CLP [3] and was run on an IBM ThinkPad T-30. Test problems consist of

- (a) nine 4-regular graphs $g4_p$ on p nodes, for $p = 20, 30, \dots, 100$;
- (b) three 8-regular graphs $g8_p$ on p nodes, for $p = 20, 30, \dots, 40$;
- (c) the Petersen graph (*pete*);
- (d) two regular graphs on 14 and 18 vertices having overfull subgraphs (*of5_14_7* and *of7_18_9_5*);
- (e) a graph from [2] on 18 vertices and 33 edges (*jgt18*).

Graphs in (a) and (b) are randomly generated and can be colored with 4 or 8 colors respectively. It is likely that most heuristics would be able to color them optimally, but our B&C algorithms have no such heuristic, i.e. they will find a feasible solution only if the solution of the LP is integer. The remaining graphs are “Class 2” graphs, i.e. graphs G that can not be colored with $\Delta(G)$ colors.

A subgraph H of a graph G is an *overfull* subgraph if $|V(H)|$ is odd, $\Delta(H) = \Delta(G)$, and $|E(H)| > \Delta(H) \cdot (|V(H)| - 1)/2$. If G has an overfull subgraph, then G is a Class 2 graph. Graphs in (d) were randomly generated and have overfull subgraphs, but are not overfull themselves. The graph in (e) is a small non-regular Class 2 graph.

To illustrate the benefits and trade-offs between the different types of cuts, we report results of three B&C algorithms. The separation algorithms for the different types of cuts are: the separation heuristic for GBI of Section 1, the exact LPC separation algorithm alluded to at the end of Section 2, the heuristic separation for MI algorithm of Section 3 (except that blocks are not computed), and the separation algorithm for SWI of Section 4. The generation of the cuts is done as follows: GBI and LPC are generated as long as possible. MI, SWI and Gomory cuts are generated only during the first six rounds of cutting, but SWI are generated only if no GBI are found (at most one round of SWI are used at each node). Finally, LPC are generated only when no GBI, MI and SWI are found.

B&C 1 uses GBI, MI, and Gomory cuts. B&C 2 uses, in addition, LPC, and B&C 3 uses all five types of cuts. Table 1 gives the number of nodes in the enumeration tree. As expected, in general, the number of nodes is smaller when more cuts are in use, but for some problems, there is a big drop between variant 1 and 2, i.e. the use of LPC seems to be important. Most of these problems have relatively large degree, which is also expected, as GBI give a good approximation of the all different polytope when the number of edges is small. On the other hand, the use of SWI does not seem to help much on these problems.

Table 2 shows that for problem with low maximum degree, using SWI increases the overall cpu time. This (and Table 3) illustrates the difficulties for separating these inequalities efficiently. Even with the restricted use of one round of SWI cuts at most, the separation algorithm returns a large number of violated SWI cuts. A better understanding of these cuts might help generate “useful” ones more efficiently. The separation times are very small for GBI, MI, and Gomory cuts. The LPC, however take a significant time (about 30% of the total time for the 8-regular graphs and 50% for of7_18_9_5). The SWI separation is also time consuming, taking roughly 10% of the total time.

References

1. David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65(1-3):21–46, 1996. First International Colloquium on Graphs and Optimization (GOI), 1992 (Grimenz).
2. Amanda G. Chetwynd and Robin J. Wilson. The rise and fall of the critical graph conjecture. *J. Graph Theory*, 7(2):153–157, 1983.
3. COIN-OR. www.coin-or.org , 2003.
4. Don Coppersmith and Jon Lee. Parsimonious binary-encoding in integer programming. *IBM Research Report RC23838*, 2003.
5. Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, second edition, 1993.

Table 1. Number of nodes.

	1	2	3
<i>g4_20</i>	13	17	21
<i>g4_30</i>	35	31	27
<i>g4_40</i>	41	45	35
<i>g4_50</i>	55	53	59
<i>g4_60</i>	61	63	67
<i>g4_70</i>	69	75	89
<i>g4_80</i>	75	83	81
<i>g4_90</i>	87	93	93
<i>g4_100</i>	107	125	105
<i>g8_20</i>	435	93	93
<i>g8_30</i>	7113	155	139
<i>g8_40</i>	5221	191	185
<i>pete</i>	5	7	7
<i>of5_14_7</i>	1537	955	879
<i>of7_18_9_5</i>	25531*	8703	7377
<i>jgt18</i>	1517	1453	1263

Table 2. cpu time in seconds. A star denotes a problem not solved in 1hr.

	1	2	3
<i>g4_20</i>	0.50	1.00	1.60
<i>g4_30</i>	1.10	1.70	2.70
<i>g4_40</i>	1.90	3.50	5.30
<i>g4_50</i>	4.50	5.70	15.40
<i>g4_60</i>	5.90	8.70	28.20
<i>g4_70</i>	5.90	11.80	39.90
<i>g4_80</i>	7.60	14.90	43.70
<i>g4_90</i>	11.50	17.80	50.50
<i>g4_100</i>	12.10	30.10	82.60
<i>g8_20</i>	67.00	48.60	58.10
<i>g8_30</i>	*	130.10	151.60
<i>g8_40</i>	3366.20	244.90	320.60
<i>pete</i>	0.30	0.50	0.50
<i>of5_14_7</i>	103.30	107.20	111.40
<i>of7_18_9_5</i>	*	*	*
<i>jgt18</i>	107.50	126.20	139.70

6. Ian Holyer. The NP-completeness of edge-coloring. *SIAM J. Comput.*, 10(4):718–720, 1981.
7. Jon Lee. All-different polytopes. *J. Comb. Optim.*, 6(3):335–352, 2002. New approaches for hard discrete optimization (Waterloo, ON, 2001).

Table 3. Number of generated cuts.

	1			2			3					
	GBI	MI	GOM	GBI	LPC	MI	GOM	GBI	LPC	MI	SWI	GOM
<i>g4_20</i>	72	2	22	124	0	6	31	102	0	0	631	37
<i>g4_30</i>	214	1	51	208	0	0	48	270	0	3	836	61
<i>g4_40</i>	282	3	70	328	0	1	70	316	0	4	1347	71
<i>g4_50</i>	524	13	102	598	0	7	96	850	0	15	4521	145
<i>g4_60</i>	770	4	132	688	0	8	128	1110	0	20	7058	173
<i>g4_70</i>	1038	10	141	971	0	16	151	1692	0	9	10169	232
<i>g4_80</i>	932	5	142	860	0	9	156	1780	0	16	9405	199
<i>g4_90</i>	1150	10	168	1054	74	7	168	1680	0	10	10560	206
<i>g4_100</i>	1414	4	183	2206	0	35	286	2260	0	8	16668	256
<i>g8_20</i>	10313	115	1248	2225	1349	23	216	2369	1460	20	3095	231
<i>g8_30</i>	338022	2633	24651	3934	2408	29	366	4036	2291	24	7236	352
<i>g8_40</i>	123843	2179	15527	4204	2820	31	441	5535	3043	39	11492	462
<i>pete</i>	0	8	19	0	0	8	19	0	0	8	0	19
<i>of5_14_7</i>	19838	680	4063	14690	3795	798	2995	14885	3305	777	8721	2899
<i>of7_18_9_5</i>	594021	12510	76323	286893	141606	7563	28801	254749	121398	7381	236250	25715
<i>jgt18</i>	13664	793	4305	13594	167	722	4138	12449	30	745	27714	3949

8. Jon Lee, Janny Leung, and Sven de Vries. Separating type-I odd-cycle inequalities for a binary encoded edge-coloring formulation. *IBM Research Report RC22303*, 2002. to appear in: *Journal of Combinatorial Optimization*.
9. R. Kipp Martin. Using separation algorithms to generate mixed integer model reformulations. *Oper. Res. Lett.*, 10(3):119–128, 1991.
10. Vijay V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph maximum matching algorithm. *Combinatorica*, 14:71–109, 1994.
11. Vadim G. Vizing. On an estimate of the chromatic class of a p -graph. *Diskret. Analiz No.*, 3:25–30, 1964.

Single Machine Scheduling with Precedence Constraints

Extended Abstract

José R. Correa and Andreas S. Schulz

Operations Research Center, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA
{jcorrea,schulz}@mit.edu

Abstract. We discuss the problem of sequencing precedence-constrained jobs on a single machine to minimize the average weighted completion time. This problem has attracted much attention in the mathematical programming community since Sidney's pioneering work in 1975. We look at the problem from a polyhedral perspective and uncover a relation between Sidney's decomposition theorem and different linear programming relaxations. More specifically, we present a generalization of Sidney's result, which particularly allows us to reason that virtually all known 2-approximation algorithms comply with his decomposition. Moreover, we establish a connection between the single-machine scheduling problem and the vertex cover problem. Indeed, in the special case of series-parallel precedence constraints, we prove that the sequencing problem can be seen as a special case of vertex cover. We also argue that this result is true for general precedence constraints if one can show that a certain integer program represents a valid formulation of the sequencing problem. Finally, we provide a characterization of the active inequalities of a linear programming relaxation in completion time variables.

1 Introduction

We consider the following scheduling problem. A set of jobs has to be processed on a single machine, which can handle at most one job at a time. Each job has a positive processing time and a nonnegative weight, and we want to find a schedule of the jobs that minimizes the weighted sum of job completion times. In its simplest form, the problem can be solved efficiently using Smith's rule [23], which sequences the jobs in nonincreasing order of their ratios of weight to processing time. The problem becomes hard when side constraints such as precedence constraints or release dates are introduced. Here, we focus on the case when the jobs have to comply with given precedence constraints. Lawler [12] and Lenstra and Rinnooy Kan [13] showed that this problem is strongly NP-hard.

Let us denote the set of jobs by $N = \{1, \dots, n\}$, their weights by $w_i \geq 0$, and their processing times by $p_i > 0$. The precedence constraints are given in the form of a directed acyclic graph $G = (N, P)$, where $(i, j) \in P$ implies that job i must be processed before job j . We assume that G is transitively closed;

i.e., if $(i, j) \in P$ and $(j, k) \in P$, then $(i, k) \in P$. The goal is to find a feasible ordering of the jobs in N such that the sum of weighted completion times is minimized. Hence, if C_i denotes the time at which job i is completed, we want to minimize $\sum_{i \in N} w_i C_i$. In standard scheduling notation [7], this problem is known as $1|prec|\sum w_j C_j$.

Several integer programming formulations as well as linear programming relaxations have been proposed for this problem. They can basically be divided into three groups according to the decision variables used: some formulations exploit time-indexed variables (e.g., [4]), which are binary variables indicating when a job is starting, completed, or active; others make direct use of completion time variables (e.g., [1]); and the third category borrows its decision variables from the underlying linear ordering polytope (e.g., [26]). We refer to [19] for an overview and a collection of further references.

Relaxations in all these variables have been successfully used to obtain constant factor approximation algorithms for this problem. The first one, proposed by Hall, Shmoys, and Wein [8], relies on a time-indexed linear programming relaxation and has performance guarantee $4 + \varepsilon$. Subsequently, Schulz [21] presented a 2-approximation algorithm based on solving a weaker linear programming relaxation in completion time variables; see also [9]. The analysis also implies that Potts' [17] linear programming relaxation in linear ordering variables can be used to obtain another 2-approximation algorithm. Later, Chudak and Hochbaum [3] proposed a relaxation of Potts' linear program that still suffices to get yet another approximation algorithm of the same performance guarantee. Moreover, they showed that the weaker linear program can be solved by one min-cut computation, which yields a combinatorial 2-approximation algorithm. Chekuri and Motwani [2] as well as Margot, Queyranne and Wang [14] independently used Sidney's decomposition theorem [22] to give a whole class of combinatorial 2-approximation algorithms. Eventually, Goemans and Williamson [6] revived two-dimensional Gantt charts [5] as a useful tool to illustrate, among other aspects, the findings of Margot et al. and Chekuri and Motwani.

One of the goals of this paper is to show that the above mentioned 2-approximation algorithms are closely related to one another. In fact, we prove that each of them follows Sidney's decomposition. In particular, after setting the stage in Section 2, we use Section 3.1 to establish a connection between Sidney's decomposition theorem and linear programming relaxations in linear ordering variables. In Section 3.2, we propose a new relaxation in linear ordering variables that suggests a strong relationship between $1|prec|\sum w_j C_j$ and the vertex cover problem. Sections 3.3 and 3.4 consider particular classes of precedence constraints. For instance, if the precedence constraints are series-parallel, the sequencing problem is indeed a special case of the vertex cover problem. We study completion time relaxations in Section 4, extending a result by Margot et al. [14] on the structure of optimal solutions. The results in Sections 3.1 and 4 imply that all known 2-approximation algorithms follow Sidney's decomposition and are therefore special cases of the class of algorithms described in [2,14].

2 Definitions and Preliminaries

For a job $i \in N$, we denote the ratio w_i/p_i by ρ_i . We generalize these quantities to sets $S \subseteq N$ of jobs in the usual way: $p(S) := \sum_{i \in S} p_i$, $w(S) := \sum_{i \in S} w_i$, and $\rho(S) := w(S)/p(S)$. For $S \subseteq N$, G_S denotes the subgraph of G induced by S , and $P(G_S)$ is the set of arcs (precedence constraints) in G_S . A set of jobs $I \subseteq S$ is called *initial* in G_S if it satisfies: $i \in I \Rightarrow j \in I$ for all $(j, i) \in P(G_S)$. Similarly, $F \subseteq S$ is called *final* in G_S if the following holds: $i \in F \Rightarrow j \in F$ for all $(i, j) \in P(G_S)$. We simply say that $S \subseteq N$ is initial (resp. final) if S is initial in G (resp. final in G). A set $S^* \subseteq S$ is a ρ -maximal initial set in G_S if and only if S^* is an initial set in G_S with maximum value of ρ . In other words, $S^* \in \arg \max\{\rho(S') : S' \text{ initial in } G_S\}$. An initial set $S \subseteq N$ is said to be *non-Sidney-decomposable* if the only ρ -maximal initial set contained in S is S itself; i.e., S is a minimal (with respect to inclusion) ρ -maximal initial set.

We are now ready to introduce the concept of a Sidney decomposition. Consider a partition of N into disjoint sets $(P_i)_{i=1}^k$ such that P_i is non-Sidney-decomposable in $P_i \cup \dots \cup P_k = N \setminus \{P_1 \cup \dots \cup P_{i-1}\}$. The collection $(P_i)_{i=1}^k$ is called a *Sidney decomposition* of N . Sidney [22] proved that there exists an optimal solution to $1|prec|\sum w_j C_j$ that processes the jobs in P_i before the ones in P_j , whenever $i < j$. This result is known as *Sidney's decomposition theorem*. Naturally, a scheduling algorithm is said to comply with Sidney's decomposition if for some Sidney decomposition $(P_i)_{i=1}^k$, the algorithm processes the jobs in P_i before the ones in P_j , whenever $i < j$. Sidney decompositions can be computed in polynomial time [12,16,14].

Lemma 1 (Sidney [22]). *If S is a ρ -maximal initial set, F is final in G_S , and I is initial in $G_{N \setminus S}$, then $\rho(I) \leq \rho(S) \leq \rho(F)$.*

Proof. It suffices to note that for two disjoint sets of jobs A and B , $\rho(A \cup B)$ can be written as a convex combination of $\rho(A)$ and $\rho(B)$. Indeed, $\rho(A \cup B) = \frac{w(A)+w(B)}{p(A)+p(B)} = \rho(A)\frac{p(A)}{p(A \cup B)} + \rho(B)\frac{p(B)}{p(A \cup B)}$. \square

A direct consequence of this lemma is that a non-Sidney-decomposable set is always connected.

We now present the linear programming relaxations for $1|prec|\sum w_i C_j$ used later in the paper. The first, due to Potts [17], is based on an integer programming formulation using linear ordering variables.

$$[P] \quad \min \sum_{i,j \in N} p_i w_j \delta_{ij} + \sum_{j \in N} p_j w_j$$

$$\delta_{ij} + \delta_{ji} = 1 \quad \text{for all } i, j \in N, \tag{1}$$

$$\delta_{ij} + \delta_{jk} + \delta_{ki} \geq 1 \quad \text{for all } i, j, k \in N, \tag{2}$$

$$\delta_{ij} = 1 \quad \text{for all } (i, j) \in P, \tag{3}$$

$$\delta_{ij} \in \{0, 1\}. \tag{4}$$

To simplify notation, we implicitly assume that this and future formulations do not contain variables of the form δ_{ii} , for $i \in N$. It is easy to see that any feasible solution δ to the above integer program represents a valid schedule. Later on, Chudak and Hochbaum [3] proposed the following relaxation of [P].

$$[\text{CH}] \quad \min \sum_{i,j \in N} p_i w_j \delta_{ij} + \sum_{j \in N} p_j w_j$$

subject to (1), (3) and (4), but relaxing (2) to

$$\delta_{ik} + \delta_{kj} \geq 1 \quad \text{for all } (i, j) \in P, k \in N. \quad (5)$$

This new integer program leads to two natural questions: Is an optimal solution of [P] optimal for [CH], too? In other words, is [CH] a valid formulation of the scheduling problem (in the sense that it gives the right objective function value and there exists an optimal solution that is a schedule)? Moreover, if [P-LP] and [CH-LP] are the linear programming relaxations of [P] and [CH], respectively, is an optimal solution to [P-LP] optimal for [CH-LP]? Let us formulate these questions as conjectures:

Conjecture 2. An optimal solution to [P] is also optimal for [CH].

Conjecture 3. An optimal solution to [P-LP] is also optimal for [CH-LP].

The conjectures are true if the set of precedence constraints is empty (see, e.g., [19]) or series-parallel (see Theorem 10 below). If true in general, they would lead to several interesting consequences, as we will see in the remainder of this paper. Moreover, we will also prove a number of results that seem to support these conjectures.

Let us also introduce a linear programming relaxation in completion time variables.

$$[\text{QW-LP}] \quad \min \sum_{j \in N} w_j C_j$$

$$\sum_{j \in S} p_j C_j \geq \frac{1}{2}(p(S)^2 + p^2(S)) \quad \text{for all } S \subseteq N, \quad (6)$$

$$C_k \geq C_j + p_k \quad \text{for all } (j, k) \in P. \quad (7)$$

Inequalities (6) are known as the *parallel inequalities*; they suffice to describe the scheduling polytope in the absence of precedence constraints [25,18]. Inequalities (7) model the precedence constraints.

Finally, we briefly describe the known classes of approximation algorithms for $1| \text{prec} | \sum w_j C_j$ with a performance guarantee of 2:

- (A) Let C be an optimal solution to [QW-LP]. Schedule the jobs in nondecreasing order of C_j , breaking ties arbitrarily [21,9].

- (B) Let δ be an optimal solution to [CH-LP]. Compute $C_j = \sum_i p_i \delta_{ij} + p_j$ and schedule the jobs in nondecreasing order of C_j , breaking ties arbitrarily [3].
- (C) Determine a Sidney decomposition of N . Schedule the jobs in any feasible order that follows this decomposition [2,14].

3 Linear Ordering Relaxations

In this section, we consider a variety of formulations and linear programming relaxations of $1| \text{prec} | \sum w_j C_j$ in linear ordering variables. In Section 3.1, we prove a structural property of the optimal solutions of [CH-LP] and [P-LP] as well as [CH] and [P], which generalizes Sidney's decomposition theorem. We propose a new linear programming relaxation in Section 3.2; while it is equivalent to [CH-LP], it helps to uncover the connection to the vertex cover problem. We study special classes of precedence constraints in Sections 3.3 and 3.4.

3.1 A Structural Result

Theorem 4. *Let S be a ρ -maximal initial set of jobs. Then there exists an optimal solution δ^* to [CH-LP] such that $\delta_{ij}^* = 1$ for all $i \in S, j \notin S$.*

Proof. Let δ^* be an optimal solution to [CH-LP]. If δ^* does not have the desired property, we can apply the following procedure. For each $k \in S$, define the set $I_k := \{i \notin S : \delta_{ik}^* > 0\}$. Similarly, for each $k \notin S$, let $F_k := \{i \in S : \delta_{ki}^* > 0\}$. Since δ^* satisfies (5), i.e., $\delta_{ik}^* + \delta_{kj}^* \geq 1$ for all $(i, j) \in P$, F_k is a final set in the graph G_S and I_k is an initial set in $G_{N \setminus S}$. Let $\varepsilon := \min\{\delta_{ij}^* : \delta_{ij}^* > 0\}$ and consider the vector δ' defined as:

$$\begin{aligned}\delta'_{ij} &:= \delta_{ij}^* + \varepsilon \quad \text{if } i \in S, j \notin S \text{ and } \delta_{ij}^* < 1; \\ \delta'_{ij} &:= \delta_{ij}^* - \varepsilon \quad \text{if } i \notin S, j \in S \text{ and } \delta_{ij}^* > 0; \\ \delta'_{ij} &:= \delta_{ij}^* \quad \text{otherwise.}\end{aligned}$$

Clearly, $0 \leq \delta'_{ij} \leq 1$ and δ' satisfies (1) and (3). For $(i, j) \in P$, (5) holds trivially when $i, j \in S$ or $i, j \notin S$. If $i \in S$ and $j \notin S$, then either δ'_{ik} or δ'_{kj} was incremented by ε and hence (5) holds. The case $j \in S, i \notin S$ does not occur since S is initial. It follows that δ' is feasible for [CH-LP].

The difference in the objective function values of δ^* and δ' can be computed as follows:

$$\begin{aligned}\sum_{i,j \in N} p_i w_j \delta_{ij}^* - \sum_{i,j \in N} p_i w_j \delta'_{ij} &= \varepsilon \sum_{k \notin S} p_k w(F_k) - \varepsilon \sum_{k \in S} p_k w(I_k) \\ &= \varepsilon \sum_{k \notin S} p_k p(F_k) \rho(F_k) - \varepsilon \sum_{k \in S} p_k p(I_k) \rho(I_k) .\end{aligned}$$

By applying Lemma 1 to the sets I_k and F_k , we obtain $\rho(I_k) \leq \rho(S) \leq \rho(F_k)$ for all $k \in N$. Hence, the above quantity can be bounded from below by:

$$\varepsilon \rho(S) \left(\sum_{k \notin S} p_k p(F_k) - \sum_{k \in S} p_k p(I_k) \right) = \varepsilon \rho(S) \left(\sum_{i \in S, j \notin S} p_i p_j - \sum_{i \in S, j \notin S} p_i p_j \right) = 0 .$$

Thus, the objective function value of δ' is not worse than that of δ^* . One can iterate this procedure with δ' until the claim is fulfilled. \square

Remark 5. The same result is true for the integer program [CH]. The proof is basically the same.

Remark 6. Theorem 4 (and the previous remark) is also valid for [P-LP] (resp. [P]). In fact, the application to the integer program [P] yields another proof of Sidney's decomposition theorem.

One consequence of Theorem 4 concerns Algorithm (B) by Chudak and Hochbaum [3]. Recall that a feasible schedule is created by first solving [CH-LP] and then sequencing the jobs according to the order defined by the projected completion times $C_j = \sum_{i \in N} p_i \delta_{ij} + p_j$.

Corollary 7. *Algorithm (B) complies with Sidney's decomposition.*

Proof. Let $(P_i)_{i=1}^k$ be a Sidney decomposition of N . Note that Theorem 4 can be iteratively applied to the sets $N, N \setminus P_1, N \setminus (P_1 \cup P_2), \dots$, which implies that there exists an optimal solution δ to [CH-LP] such that $\delta_{ij} = 1$ for all $i \in P_r$, $j \in P_s$ with $r < s$. The claim follows. \square

This corollary implies that Chudak and Hochbaum's algorithm belongs to the class of algorithms introduced by Margot, Queyranne and Wang [14], and Chekuri and Motwani [2].

3.2 A New LP-relaxation

We now propose a new linear ordering relaxation of $1|prec|\sum w_j C_j$, which can be interpreted as a vertex cover problem. We also prove that this formulation is equivalent to [CH]. The integer program is as follows:

$$[CS] \quad \min \sum_{i,j \in N} p_i w_j \delta_{ij} + \sum_{j \in N} p_j w_j$$

subject to (5), (4), (3), $\delta_{ij} = 0$ for $(j, i) \in P$, and

$$\delta_{ij} + \delta_{ji} \geq 1 \quad \text{for all } i, j \in N, \tag{8}$$

$$\delta_{il} + \delta_{kj} \geq 1 \quad \text{for all } (i, j), (k, l) \in P. \tag{9}$$

As usual, let us denote by [CS-LP] the linear relaxation of this integer program. Since we can obviously omit the variables δ_{ij} and δ_{ji} for $(i, j) \in P$ from the formulation, [CS] is clearly a special case of the vertex cover problem. In fact, if we denote by $i \parallel j$ that neither (i, j) nor (j, i) belongs to P , it is straightforward to see that [CS-LP] is equivalent to

$$[CS'-LP] \quad \min \sum_{i,j \in N} p_i w_j \delta_{ij} + \sum_{j \in N} p_j w_j + \sum_{(i,j) \in P} p_i w_j$$

$$\delta_{ij} + \delta_{ji} \geq 1 \quad \text{for all } i, j \in N, i \parallel j, \quad (10)$$

$$\delta_{ik} + \delta_{kj} \geq 1 \quad \text{for all } (i, j) \in P, k \in N, i \parallel k \parallel j, \quad (11)$$

$$\delta_{il} + \delta_{kj} \geq 1 \quad \text{for all } (i, j), (k, l) \in P, i \parallel l \text{ and } j \parallel k, \quad (12)$$

$$\delta_{ij} \geq 0 \quad \text{for all } i, j \in N, i \parallel j. \quad (13)$$

Let us show next that [CS'-LP] is the dominant of [CH'-LP], where [CH'-LP] is the following linear program equivalent to [CH-LP]:

$$[\text{CH}'\text{-LP}] \quad \min \sum_{i, j \in N} p_i w_j \delta_{ij} + \sum_{j \in N} p_j w_j + \sum_{(i, j) \in P} p_i w_j$$

subject to (11), (13), and $\delta_{ij} + \delta_{ji} = 1$ for all $i, j \in N, i \parallel j$.

Lemma 8. *The optimal solutions to [CH-LP] and the optimal solutions to [CS'-LP] coincide. Moreover, [CS'-LP] is the dominant of [CH'-LP].*

Proof. Let δ be a feasible solution to [CH-LP] and let $(i, j), (k, l) \in P$. Since δ satisfies the inequalities (5), it follows that $2\delta_{kj} + 2\delta_{il} + \delta_{lj} + \delta_{jl} + \delta_{ki} + \delta_{ik} \geq 4$. As δ also satisfies (1), we can infer that δ satisfies inequality (9).

On the other hand, let δ be a feasible solution to [CS-LP] such that $\delta_{ij} + \delta_{ji} > 1$ for some $i, j \in N$. Say $\delta_{ij} = a$ and $\delta_{ji} = b$, with $a + b > 1$.

We claim that either δ_{ij} or δ_{ji} (or both) can be reduced. Suppose not. Then, both (i, j) and (j, i) must belong to a tight inequality of the form (9) or (5), respectively. There are four non-symmetric cases to be analyzed. These are presented in Figure 1, where bold-face arcs represent precedence constraints.

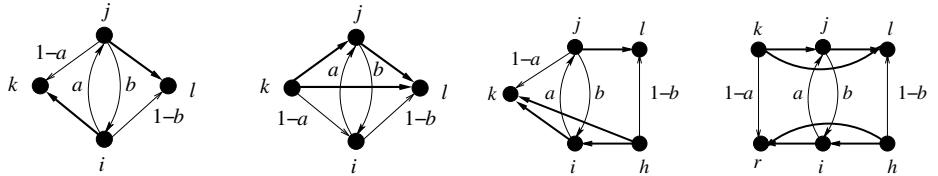


Fig. 1. Non-symmetric cases (i), (ii), (iii), and (iv)

It turns out that in all four cases a violated inequality can be found: case (i) $\delta_{jk} + \delta_{il} = 2 - (a + b) < 1$; case (ii) $\delta_{ki} + \delta_{il} = 2 - (a + b) < 1$; case (iii) $\delta_{jk} + \delta_{hl} = 2 - (a + b) < 1$; and case (iv) $\delta_{kr} + \delta_{hl} = 2 - (a + b) < 1$. As this contradicts the feasibility of δ , it follows that the values of δ_{ij} and δ_{ji} can be decreased until $\delta_{ij} + \delta_{ji} = 1$. As a result, [CS'-LP] is the dominant of [CH'-LP]. In particular, an optimal solution to [CS-LP] satisfies (1) and hence is feasible (and optimal) for [CH-LP]. \square

The previous lemma implies that Theorem 4 is also valid for [CS'-LP] and [CS]. Moreover, as [CS] represents an instance of the vertex cover problem, it

follows from the work of Nemhauser and Trotter [15] that [CS-LP] is half-integral and that an optimal LP solution can be obtained via a single min-cut computation. Hence, the same holds for [CH-LP], which implies the main result in Chudak and Hochbaum's paper [3, Theorem 2.4].

Interestingly, Theorem 4 also implies that we can (possibly) get a refinement of a Sidney decomposition as follows:

- (a) Perturb the job weights such that [CS-LP] has a unique optimal solution.
- (b) Let δ be that optimal solution.
- (c) Consider the digraph $D = (N, A)$ with arc set $A := \{(i, j) : \delta_{ij} = 1\}$.
- (d) Remove cycles from D .
- (e) Compute the series decomposition (N_1, N_2, \dots, N_q) of the remaining digraph.

It follows from Theorem 4 that (N_1, N_2, \dots, N_q) is a refinement of a Sidney decomposition. In particular, every feasible job sequence that follows this order is a 2-approximation.

In this light, it is of course tempting to conjecture that what the Sidney decomposition does for the scheduling problem is indeed equivalent to what the persistency property [15] implies for the vertex cover problem. In particular, we know that every feasible schedule that complies with a Sidney decomposition is a 2-approximation for the scheduling problem [2,14], while every feasible vertex cover that includes all variables that are equal to one in an optimal basic feasible solution to the vertex cover LP is a 2-approximation for the vertex cover problem [10]. In fact, one might conjecture that the following is true:

Let δ be a unique optimal basic feasible solution to [CH-LP]. Then there exists an optimal schedule in which job i precedes job j whenever $\delta_{ij} = 1$.

However, short of a proof of Conjecture 2, we have to confine ourselves to the following result.

Lemma 9. *If δ is an optimal solution to [CS-LP], then there exists an optimal solution δ' to its integer counterpart [CS] such that $\delta'_{ij} = 1$ whenever $\delta_{ij} = 1$. Moreover, $\delta'_{ij} + \delta'_{ji} = 1$ for all $i, j \in N$.*

Let us mention some additional consequences of Conjecture 2, if it were true. Most importantly, Lemma 8 would immediately imply that $1|\text{prec}| \sum w_j C_j$ is a special case of the vertex cover problem. Combined with Lemma 9, this would also mean that any approximation algorithm for vertex cover would translate into an approximation algorithm for $1|\text{prec}| \sum w_j C_j$ with the same performance guarantee.

Let us finally point out that a similar analysis to that in the proof of Lemma 8 shows that the dominant of [P-LP] is the following linear program:

$$[\text{P}'\text{-LP}] \quad \min \sum_{i,j \in N} p_i w_j \delta_{ij} + \sum_{j \in N} p_j w_j + \sum_{(i,j) \in P} p_i w_j$$

$$\begin{aligned}\delta_{ij} + \delta_{ji} &\geq 1 \quad \text{for all } i, j \in N, i \parallel j, \\ \delta(\mathcal{C}) &\geq 1 \quad \text{for all delta-cycles } \mathcal{C}, \\ \delta_{ij} &\geq 0 \quad \text{for all } i, j \in N, i \parallel j,\end{aligned}$$

where a delta-cycle $\mathcal{C} \subseteq (N \times N) \setminus P$ is a collection of arcs such that there exists a set $P_{\mathcal{C}}$ of reversed precedence constraints (i.e., a subset of $P^{-1} := \{(i, j) \in N \times N : (j, i) \in P\}$) satisfying that $\mathcal{C} \cup P_{\mathcal{C}}$ is a directed cycle in $N \times N$. [CS'-LP] is the relaxation of [P'-LP] that only considers delta-cycles of size at most 2.

Let $[\text{LP}_i]$ denote (the optimal value of) the above linear program restricted to delta-cycles of size no more than i . Then $[\text{LP}_2] = [\text{CS}'\text{-LP}]$ and $[\text{LP}_{\infty}] = [\text{P}'\text{-LP}]$; moreover

$$[\text{LP}_2] \leq [\text{LP}_3] \leq \cdots \leq [\text{LP}_k] \leq \cdots \leq [\text{LP}_{\infty}].$$

Conjecture 3 states that this chain of inequalities is actually a chain of equalities.

3.3 Series-Parallel Precedence Constraints

Even though $1|\text{prec}| \sum w_j C_j$ is strongly NP-hard, some special cases can be solved efficiently. For example, Lawler [12] presented a polynomial-time algorithm for series-parallel precedence constraints. Moreover, Queyranne and Wang [20] gave a complete characterization of the convex hull of feasible completion time vectors, while Goemans and Williamson [6] proposed a primal-dual algorithm that unifies both results.

Series-parallel precedence constraints can be defined inductively; the base elements are individual jobs. Given two series-parallel digraphs $G_1 = (N_1, P_1)$ and $G_2 = (N_2, P_2)$ such that $N_1 \cap N_2 = \emptyset$, the parallel composition of G_1 and G_2 results in a partial order on $N_1 \cup N_2$ that maintains P_1 and P_2 , but does not introduce any additional precedence relationships. The series composition of G_1 and G_2 leads to a partial order on $N_1 \cup N_2$ that maintains P_1 and P_2 ; moreover, if $i \in N_1$ and $j \in N_2$, then i precedes j in the new partial order.

With Theorem 4 in place, the proof of the following result becomes remarkably simple.

Theorem 10. *When the precedence constraints are series-parallel, [CH-LP] has an integer optimal solution that is a feasible schedule.*

Proof. We proceed by induction on the number of jobs. The result is trivial when $|N| = 1$ or $|N| = 2$. Assume the result holds for all sets N of jobs with series-parallel precedence constraints such that $|N| \leq n$. Note that if (N, P) is series-parallel, then any induced subgraph is series-parallel, too. Let us consider a set N of jobs with $|N| = n + 1$:

- (i) If $G = (N, P)$ has a series decomposition $G = G_1 \rightarrow G_2$, then $|N_1|, |N_2| \leq n$ and the induction hypothesis applies to N_1 and N_2 . Since the values δ_{ij} for $i \in N_1$ and $j \in N_2$ are fixed by the decomposition, the result holds for N .

- (ii) Otherwise, $N = N_1 \cup N_2$ and $i \parallel j$ for all $i \in N_1$ and $j \in N_2$. Clearly, there is a non-Sidney-decomposable set that is either fully contained in N_1 or in N_2 [22]. Let S be this set. By Theorem 4, there is an optimal solution satisfying $\delta_{ij} = 1$ for all $i \in S$, $j \in N \setminus S$. Hence, we obtain the desired result by applying the inductive hypothesis to S and $N \setminus S$. \square

In particular, $1|\text{prec}|\sum w_j C_j$ is a special case of the vertex cover problem for series-parallel precedence constraints. Moreover, an optimal schedule can be computed by a single min-cut computation, which gives another polynomial-time algorithm for the case of series-parallel precedence constraints.

3.4 A Larger Class of Precedence Constraints Potentially Solvable in Polynomial Time

Let P be the set of precedence constraints on the job set N . A linear extension L of P over N is a total ordering (tournament) of the elements in N such that $(i, j) \in P \Rightarrow (i, j) \in L$. We use the notation $i <_P j$ to indicate that i is an immediate predecessor of j in P . That is, there is no $l \in N \setminus \{i, j\}$ such that $(i, l), (l, j) \in P$.

Definition 11. A set P of precedence constraints is simple if there exists a linear extension L of P such that for all $i <_P j$ and for all $i \parallel k \parallel j$, either $(k, i), (k, j) \in L$ or $(i, k), (j, k) \in L$.

It is easy to see that simple orders contain, among others, series-parallel orders. In fact, simple orders coincide with orders of dimension 2.

Proposition 12. The class of 2-dimensional orders is identical to the class of simple orders.

We omit the proof from this extended abstract. Kolliopoulos and Steiner [11] gave a $(1/2(\sqrt{5}+1)+\varepsilon)$ -approximation algorithm for the single-machine scheduling problem with 2-dimensional precedence constraints, using machinery developed by Woeginger [24].

Theorem 13. If P is a simple order, then every basic feasible solution to [CH-LP] is integer.

Proof. First, note that in [CH-LP], it is enough to write inequalities (5) for $(i, j) \in P$ such that $i <_P j$. Consider the ordering L ; we can then rewrite [CH-LP] as follows:

$$\begin{aligned} [\text{CH-LP}] \quad \min \sum_{(i,j) \in L} [p_i w_j \delta_{ij} + p_j w_i (1 - \delta_{ij})] + \sum_{j \in N} p_j w_j + \sum_{(i,j) \in P} p_i w_j \\ \delta_{jk} - \delta_{ik} \leq 0 \quad \text{for all } i <_P j, (i, k), (j, k) \in L, \\ \delta_{ki} - \delta_{kj} \leq 0 \quad \text{for all } i <_P j, (k, i), (k, j) \in L, \\ -\delta_{ij} \leq 0 \quad \text{for all } (i, j) \in L, \\ \delta_{ij} \leq 1 \quad \text{for all } (i, j) \in L. \end{aligned}$$

For $(i, j) \in L$, the variable δ_{ji} does not appear. It remains to be observed that the constraint matrix of this linear program is totally unimodular. \square

Given this theorem, it is obvious that the correctness of Conjecture 2 (or of Conjecture 3) would imply that $1|prec|\sum w_j C_j$ is solvable in polynomial time for simple (i.e., 2-dimensional) precedence constraints.

4 Structure of the LP in Completion Time Variables

In this section, we study properties of the [QW-LP] relaxation of $1|prec|\sum w_j C_j$. In particular, we show that Algorithm (A) complies with Sidney's decomposition as well. Moreover, we relate the structure of the optimal solutions to this linear program to Sidney's decomposition theorem, thereby extending a result of Margot et al. [14]. In fact, we show that in an optimal solution of [QW-LP], the tight parallel inequalities are exactly those corresponding to the sets in a Sidney decomposition.

Borrowing notation from [14], we define the family of tight sets associated with an optimal solution C^* to [QW-LP] as

$$\tau(C^*) = \{J : J \subseteq N \text{ and inequality (6) is tight for } J\}.$$

By convention, we assume that $\emptyset \in \tau(C^*)$. The following lemma combines Lemmas 4.1 and 4.2 in [14].

Lemma 14 (Margot, Queyranne, and Wang [14]). *Let C^* be an optimal solution to [QW-LP]. Let $J \in \tau(C^*)$ be a non-trivial tight set.*

- (i) *For all jobs $i \in J$, $C_i^* \leq p(J)$, and equality holds if and only if $J \setminus \{i\} \in \tau(C^*)$.*
- (ii) *For all jobs $i \notin J$, $C_i^* \geq p(J) + p_i$, and equality holds if and only if $J \cup \{i\} \in \tau(C^*)$.*
- (iii) *If there is no tight set $\hat{J} \subset J$ with $|\hat{J}| = |J| - 1$, then $C_j^* - C_i^* > p_j$ for all $i \in J, j \notin J$.*
- (iv) *If there is no tight set $\hat{J} \supset J$ with $|\hat{J}| = |J| + 1$, then $C_j^* - C_i^* > p_j$ for all $i \in J, j \notin J$.*

Throughout this section we assume that C^* is an optimal solution to [QW-LP] and that $C_1^* \leq C_2^* \leq \dots \leq C_n^*$. Let $A, B \subseteq N = \{1, \dots, n\}$ such that $A \cap B = \emptyset$. For $\varepsilon > 0$, we define the perturbed completion time vector $C^\varepsilon(A, B)$ as follows:

$$\begin{aligned} C_i^\varepsilon(A, B) &= C_i^* + \frac{\varepsilon}{p(A)} \quad \text{for all } i \in A, \\ C_i^\varepsilon(A, B) &= C_i^* - \frac{\varepsilon}{p(B)} \quad \text{for all } i \in B, \text{ and} \\ C_i^\varepsilon(A, B) &= C_i^* \quad \text{for all other jobs.} \end{aligned}$$

Lemma 15. Let $A, B \subseteq N$ such that $i < j$ for all $i \in A, j \in B$. Then, for ε small enough, $C_i^\varepsilon(A, B)$ satisfies (6), and

$$\sum_{j \in N} w_j C_j^* - \sum_{j \in N} w_j C_j^\varepsilon(A, B) = \varepsilon(\rho(B) - \rho(A)) .$$

Proof. From (i) and (ii) in Lemma 14, we know that sets in $\tau(C^*)$ are of the form $\{1, \dots, t\}$. Therefore, it is enough to check that $C^\varepsilon(A, B)$ satisfies all inequalities (6) corresponding to sets of the form $\{1, \dots, t\}$. The result is obvious if $t < \min\{i : i \in B\}$, so let us check the result only for larger t . With $B' := B \cap \{1, \dots, t\}$, $p(B') \leq p(B)$ and

$$\begin{aligned} \sum_{j=1}^t p_j C_j^\varepsilon(A, B) &= \sum_{j=1}^t p_j C_j^* + \varepsilon \frac{p(A)}{p(A)} - \varepsilon \frac{p(B')}{p(B)} \\ &\geq \sum_{j=1}^t p_j C_j^* \geq \frac{1}{2}(p(\{1, \dots, t\})^2 + p^2(\{1, \dots, t\})) . \end{aligned}$$

Finally, a straightforward calculation shows that the difference in the objective function values is exactly $\varepsilon(\rho(B) - \rho(A))$. \square

Lemma 16. If S is a ρ -maximal initial set, then there exists an optimal solution C^* to [QW-LP] in which $C_i^* \leq C_j^*$ for all $i \in S, j \notin S$.

Proof. Let C^* be an optimal solution to [QW-LP]. W.l.o.g., assume that C^* is the unique optimum. Consider a ρ -maximal initial set S and suppose that in the sequence defined by C^* there exists jobs $l \notin S$ and $k \in S$ such that $C_l^* < C_k^*$. Let us assume that k is the job in S of maximum C_k^* , i.e., $k \in \arg \max\{C_i^* : i \in S\}$. Let l_1, l_2, \dots, l_u be the jobs in $N \setminus S$ in nondecreasing order of completion time and such that $C_{l_1}^* \leq \dots \leq C_{l_u}^* \leq C_k^*$.

From (i) and (ii) in Lemma 14, we know that the only candidate sets in $\tau(C^*)$ are of the form $\{1, \dots, i\}$. We distinguish three possible cases and, for each case, we use Lemma 15 to derive a contradiction.

Case 1. For some r with $l_1 \leq r < k$, $\{1, \dots, r\} \in \tau(C^*)$, and either $\{1, \dots, r-1\} \notin \tau(C^*)$ or $\{1, \dots, r+1\} \notin \tau(C^*)$.

Suppose r is such that $l_1 < l_v \leq r < l_{v+1} < l_u$. In this case, let $A = \{l_1, \dots, l_v\}$ and $B = S \cap \{r+1, \dots, k\}$ and consider the perturbed solution $C^\varepsilon(A, B)$.

We prove now that for sufficiently small ε , $C^\varepsilon(A, B)$ satisfies (6) and (7). The former is implied by Lemma 15. To see the latter, remember that S is an initial set. Moreover, inequality (6) is tight for $\{1, \dots, r\}$. This, together with Lemma (14), implies that $C^\varepsilon(A, B)$ satisfies (7).

Finally, since B is a final set in G_S and A is an initial set in $G_{N \setminus S}$, the difference in the objective function values is $\varepsilon(\rho(B) - \rho(A)) \geq 0$. Thus, $C^\varepsilon(A, B)$ is another optimal solution to [QW-LP].

Case 2. For all $l_1 \leq i < k$, $\{1, \dots, i\} \notin \tau(C^*)$.

Let $\{1, \dots, r\}$ and $\{1, \dots, s\}$ be the largest set in $\tau(C^*)$ not containing l_1 and the smallest set in $\tau(C^*)$ containing $k - 1$, respectively. Let $A = \{i : i \notin S, r < i \leq s\}$ and $B = S \cap \{r + 1, \dots, k\}$, and consider again the perturbation $C^\varepsilon(A, B)$.

It is clear that, for small ε , $C^\varepsilon(A, B)$ satisfies (6) (by Lemma 15) and (7) (by Lemma 14). As in the previous case, the difference in the objective function value is nonnegative, since B is final in G_S and A is initial in $G_{N \setminus S}$.

Case 3. For all $l_1 \leq i < k$, $\{1, \dots, i\} \in \tau(C^*)$.

Let $s \in S$ be the job with the smallest completion time (i.e., smallest index) satisfying $C_s^* > C_{l_1}^*$. Let $A = \{l_1, \dots, s - 1\}$ and $B = S \cap \{s, \dots, k\}$. The rest of the argument is similar to the previous cases. \square

As described earlier, Algorithm (A) first solves [QW-LP] and then processes the jobs in the order defined by the optimal LP solution $C_1^* \leq C_2^* \leq \dots \leq C_n^*$. The last lemma implies that the resulting schedule follows Sidney's decomposition. Hence, all known 2-approximation algorithms for $1|prec| \sum w_j C_j$ preserve Sidney's decomposition; therefore, they belong to the class of algorithms studied in [2,14], which already implies their performance ratio.

Lemma 17. *Let C^* be an optimal solution to [QW-LP] and let $\{P_1, P_2, \dots, P_k\}$ be a Sidney decomposition of N . Then inequality (6) is tight for the sets $P_1, P_1 \cup P_2, \dots, P_1 \cup \dots \cup P_k$.*

Proof. We already know that $C_r^* \leq C_s^*$ if $r \in P_i$ and $s \in P_j$ with $i < j$. Assume that $P_1 \notin \tau(C^*)$ and let $\{1, \dots, r\}$ and $\{1, \dots, s\}$ be the biggest and smallest sets in $\tau(C^*)$ contained in and containing P_1 , respectively. We can define a new feasible solution $C^\varepsilon(A, B)$ to [QW-LP] by taking $A = (N \setminus P_1) \cap \{r + 1, \dots, s\}$ and $B = P_1 \cap \{r + 1, \dots, s\}$. Using Lemma 15 it is easy to check that $C^\varepsilon(A, B)$ is a feasible solution with cost smaller than the cost of C^* . The lemma follows inductively for the rest of the sets in the Sidney decomposition. \square

We now show that these sets are essentially the only tight sets in an optimal solution to [QW-LP], which was proved earlier by Margot et al. [14] when N is non-Sidney-decomposable.

Let $\{Q_1, Q_2 \setminus Q_1, \dots, Q_k \setminus (Q_1 \cup \dots \cup Q_{k-1})\}$ be a Sidney decomposition of N such that $Q_i = \{1, \dots, q_i\}$ with $q_1 \leq \dots \leq q_k$, and suppose that $\{1, \dots, r\} \in \tau(C^*)$ for some $q_i < r < q_{i+1}$. If, in addition, r is such that $\{1, \dots, r+1\} \notin \tau(C^*)$, we let $A = \{q_i, \dots, r\}$ and $B = \{r + 1, \dots, q_{i+1}\}$ and consider $C^\varepsilon(A, B)$. By Lemmas 14 and 15, $C^\varepsilon(A, B)$ is feasible and its objective function value is strictly better than that of C^* . Thus all (or none of the) inequalities $\{1, \dots, r\}$ must be tight for $q_i < r < q_{i+1}$. Therefore, by combining the inequalities (6), one can easily show that

$$C_j^* = C_{q_i}^* + \sum_{l=q_i+1}^j p_l \quad \text{for all } q_i \leq j \leq q_{i+1} .$$

This implies that $Q_{i+1} \setminus Q_i$ must form a chain (again by a perturbation argument). It follows that the inequalities (6) for $\{1, \dots, r\}$ for $q_i < r < q_{i+1}$ are implied by the parallel inequalities for the sets in a Sidney decomposition and the inequalities (7). We have proved the following theorem, which characterizes the tight inequalities of an optimal solution to [QW-LP].

Theorem 18. *Let C^* be an optimal solution to [QW-LP] and let $\{P_1, P_2, \dots, P_k\}$ be a Sidney decomposition of N . Then, inequalities (6) are tight for the sets $P_1, P_1 \cup P_2, \dots, P_1 \cup \dots \cup P_k$ and all other inequalities (6) are either redundant or not tight.*

References

1. E. Balas, “On the facial structure of scheduling polyhedra”, *Mathematical Programming Study* 24, 179-218, 1985.
2. C. Chekuri & R. Motwani, “Precedence constrained scheduling to minimize sum of weighted completion time on a single machine”, *Discrete Applied Mathematics* 98, 29-38, 1999.
3. F. Chudak & D.S. Hochbaum, “A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine” *Operations Research Letters* 25, 199-204, 1999.
4. M.E. Dyer & L.A. Wolsey, “Formulating the single machine sequencing problem with release dates as a mixed integer program”, *Discrete Applied Mathematics* 26, 255-270, 1990.
5. W.L. Eastman, S. Even & I.M. Isaacs, “Bounds for the optimal scheduling of n jobs on m processors”, *Management Science* 11, 268-279, 1964.
6. M.X. Goemans & D.P. Williamson, “Two dimensional Gantt charts and a scheduling algorithm of Lawler”, *SIAM Journal on Discrete Mathematics* 13, 281-294, 2000.
7. R.L. Graham, E.L. Lawler, J.K. Lenstra & A.H.G. Rinnooy Kan, “Optimization and approximation in deterministic sequencing and scheduling: A survey”, *Annals of Discrete Mathematics* 5, 287-326, 1979.
8. L.A. Hall, D.B. Shmoys & J. Wein, “Scheduling to minimize average completion time: Off-line and on-line algorithms”, Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, 142-151, 1996.
9. L.A. Hall, A.S. Schulz, D.B. Shmoys & J. Wein, “Scheduling to minimize average completion time: Off-line and on-line approximation algorithms”, *Mathematics of Operations Research* 22, 513-544, 1997.
10. D.S. Hochbaum, “Efficient bounds for the stable set, vertex cover and set packing problems”, *Discrete Applied Mathematics* 6, 243-254, 1983.
11. S. Kolliopoulos & G. Steiner, “Partially-ordered knapsack and applications to scheduling”, R.H. Möhring, R. Raman (eds.), *Algorithms – ESA 2002*, Proceedings of the 10th Annual European Symposium on Algorithms, *Lecture Notes in Computer Science* 2461, Springer, Berlin, 612-624, 2002.
12. E.L. Lawler, “Sequencing jobs to minimize total weighted completion time subject to precedence constraints”, *Annals of Discrete Mathematics* 2, 75-90, 1978.
13. J.K. Lenstra & A.H.G. Rinnooy Kan, “Complexity of scheduling under precedence constraints”, *Operations Research* 26, 22-35, 1978.

14. F. Margot, M. Queyranne & Y. Wang, "Decompositions, network flows, and a precedence constrained single-machine scheduling problem", *Operations Research* 51, 981-992, 2003.
15. G.L. Nemhauser & L.E. Trotter, "Vertex packing: Structural properties and algorithms", *Mathematical Programming* 8, 232-248, 1978.
16. J.C. Picard, "Maximum closure of a graph and applications to combinatorial problems", *Management Science* 22, 1268-1272, 1976.
17. C.N. Potts, "An algorithm for the single machine sequencing problem with precedence constraints", *Mathematical Programming Study* 13, 78-87, 1980.
18. M. Queyranne, "Structure of a simple scheduling polyhedron", *Mathematical Programming* 58, 263-285, 1993.
19. M. Queyranne & A.S. Schulz, "Polyhedral approaches to machine scheduling", Technical Report 408, Department of Mathematics, Technische Universität Berlin, Germany, 1994.
20. M. Queyranne & Y. Wang, "Single-machine scheduling polyhedra with precedence constraints", *Mathematics of Operations Research* 16, 1-20, 1991.
21. A.S. Schulz, "Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds", W.H. Cunningham, S.T. McCormick, M. Queyranne (eds.), Proceedings of the 5th Integer Programming and Combinatorial Optimization Conference, *Lecture Notes in Computer Science* 1084, Springer, Berlin, 301-315, 1996.
22. J.B. Sidney, "Decomposition algorithms for single machine scheduling with precedence relations and deferral costs", *Operations Research* 23, 283-298, 1975.
23. W.E. Smith, "Various optimizers for single-stage production", *Naval Research and Logistics Quarterly* 3, 59-66, 1956.
24. G.J. Woeginger, "On the approximability of average completion time scheduling under precedence constraints", *Discrete Applied Mathematics* 131, 237-252, 2003.
25. L.A. Wolsey, "Mixed integer programming formulations for production planning and scheduling problems", invited talk at the 12th International Symposium on Mathematical Programming, MIT, Cambridge, MA, 1985.
26. L.A. Wolsey, "Formulating single machine scheduling problems with precedence constraints", *Economic Decision-Making: Games, Econometrics and Optimisation*, J.J. Gabszewic, J.F. Richard, L.A. Wolsey (eds.), North-Holland, Amsterdam, 1990, pp 473-484.

The Constrained Minimum Weighted Sum of Job Completion Times Problem

Asaf Levin¹ and Gerhard J. Woeginger^{2,3}

¹ Faculty of Industrial Engineering and Management, Technion
Haifa 32000, Israel

levinas@tx.technion.ac.il

² Department of Mathematics, University of Twente, P.O. Box 217
7500 AE Enschede, The Netherlands
g.j.woeginger@math.utwente.nl

³ Department of Mathematics and Computer Science, TU Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
gwoegi@win.tue.nl

Abstract. We consider the problem of minimizing the weighted sum of job completion times on a single machine (subject to certain job weights) with an additional side constraint on the weighted sum of job completion times (with respect to different job weights). This problem is NP-hard, and we provide a polynomial time approximation scheme for this problem. Our method is based on Lagrangian relaxation mixed with carefully guessing the positions of certain jobs in the schedule.

Keywords: scheduling; bicriteria optimization; approximation scheme.

1 Introduction

Consider a set of n jobs $1, 2, \dots, n$ where each job j has a positive integer processing time p_j , and two positive integer weights w_j and u_j . These jobs have to be scheduled on a single machine. A *schedule* essentially corresponds to a permutation $\pi \in S_n$ of the jobs, where $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ lists the jobs in the schedule beginning with the first one and ending with the last one. A schedule π induces for every job j a *completion time* C_j^π ; if the schedule π is clear from the context, we will suppress the superscript π and simply write C_j . In this paper, we study the following bicriteria version of the problem of minimizing the sum of weighted job completion times:

$$\begin{aligned} & \text{minimize } \sum_{j=1}^n w_j C_j^\pi \\ & \text{subject to } \sum_{j=1}^n u_j C_j^\pi \leq U \\ & \quad \pi \in S_n \end{aligned} \tag{1}$$

Here the integer U is a budget bound that is specified as part of the input. This optimization problem is called the CONstrained MINIMUM WEIGHTED SUM OF JOB COMPLETION TIMES PROBLEM ON A SINGLE MACHINE (CWSCT). It arguably belongs to the most simple non-trivial bicriteria scheduling problems.

Our Results. In Section 2 we give a straightforward reduction from the partition problem that shows that CWSCT is NP-hard. As our main result, we then construct in Section 3 a polynomial time approximation scheme (PTAS) for CWSCT: A ρ -approximation algorithm is a polynomial time algorithm that returns a near-optimal feasible solution with cost at most a factor ρ above the optimal cost. A polynomial time approximation scheme (PTAS) is a family of $(1 + \varepsilon)$ -approximation algorithms over all $\varepsilon > 0$. Our method is based on Lagrangian relaxation, on adjacency relations between various optimal permutations, and on a number of guessing steps that guess the positions of certain jobs in an optimal schedule.

Known Results. The literature contains a number of results on bicriteria scheduling problems: McCormick & Pinedo [6] consider the problem of scheduling n jobs on m uniform machines with job preemptions. They present a polynomial time algorithm that generates the entire trade-off curve of schedules which are Pareto-optimal for the makespan and the flowtime objectives. Shmoys & Tardos [9] consider a scheduling problem with n jobs and m unrelated machines. Assigning job j to machine i causes a processing time p_{ij} on this machine and also a global cost c_{ij} . For a given (hard) budget constraint C on the total cost and for a given bound T on the makespan, the algorithm in [9] either proves that there is no schedule of cost at most C and makespan at most T , or otherwise it provides a schedule with cost at most C and makespan at most $2T$. Hence, this yields a kind of 2-approximation algorithm for this budget problem. The results in [9] improve on earlier results by Lin & Vitter [5] on the same problem. Stein & Wein [11] consider a fairly general family of bicriteria scheduling problems. They show that all problems in their family possess schedules that simultaneously approximate the optimal makespan within a factor of 1.88 and the optimal sum of weighted job completion times within a factor of 1.88. Aslam, Rasala, Stein & Young [1] and Rasala, Stein, Torng & Uthaisombut [7] provide a variety of other results in this direction.

There is a rich literature on bicriteria problems in graph theory; see Ravi [8] for a (restricted) survey. Here we only want to mention a result by Goemans & Ravi [3] that yields a PTAS for the constrained minimum spanning tree problem. The approach in [3] is based on a Lagrangian relaxation approach, and it heavily exploits the adjacency structure of the underlying matroid polytope. When we started to work on CWSCT, we were hoping to be able to apply the machinery of [3] more or less directly. However, the lack of matroid properties in CWSCT seems to make this impossible; we worked around this by introducing a number of guessing steps that provide us with the necessary combinatorial structures.

2 NP-hardness

In this section we prove NP-hardness of CWSCT via a reduction from the NP-hard PARTITION problem (see Garey & Johnson [2]). An instance of PARTITION consists of n positive integers a_1, a_2, \dots, a_n such that $\sum_{i=1}^n a_i = 2A$ where A is

an integer. The problem is to decide whether there exists an index set S such that $\sum_{i \in S} a_i = \sum_{i \notin S} a_i = A$.

From an instance of PARTITION we define an instance of CWSCT as follows: Let $M = 4A^2 + 1$. There are $n + 1$ jobs $1, 2, \dots, n, n + 1$. The jobs j with $1 \leq j \leq n$ have $p_j = w_j = a_j$ and $u_j = 0$, whereas the last job has $p_{n+1} = M$, $u_{n+1} = 1$, and $w_{n+1} = 0$. Finally, the budget U equals $M + A$. We claim that the PARTITION instance has answer YES, if and only if the CWSCT instance has a schedule with objective value at most $4A^2 + MA$.

First assume that the PARTITION instance has answer YES, and that there is an index set S with $\sum_{i \in S} a_i = A$. We construct a schedule π that first runs the jobs in S (in arbitrary order), then runs job $n + 1$, and finally runs the remaining jobs (in arbitrary order). In schedule π job $n + 1$ completes exactly at time $M + \sum_{i \in S} a_i = M + A = U$. Since all other jobs have zero u -weight, the schedule π indeed satisfies the budget constraint, and hence is feasible. The total w -weight of the jobs that are processed after job $n + 1$ exactly equals A . Therefore, the objective value of π is

$$\begin{aligned} \sum_{j=1}^{n+1} w_j C_j &\leq \sum_{j=1}^{n+1} \sum_{i: \pi_i \geq \pi_j} p_j w_i \leq (\sum_{i=1}^n p_j) (\sum_{j=1}^n w_i) + p_{n+1} \sum_{i \notin S} w_i \\ &= 4A^2 + MA. \end{aligned}$$

Next, assume that the constructed CWSCT instance has a schedule with objective value at most $4A^2 + MA$. We denote by S the set of jobs in π that are processed before job $n + 1$ starts. We will show that $\sum_{i \in S} a_i = A$, and hence the PARTITION instance has answer YES. Since $C_{n+1} = \sum_j u_j C_j \leq U = M + A$ and since $C_{n+1} = p_{n+1} + \sum_{i \in S} p_i = M + \sum_{i \in S} p_i$, we conclude $\sum_{i \in S} p_i \leq A$. Therefore,

$$\sum_{i \in S} a_i \leq A. \tag{2}$$

Suppose for the sake of contradiction that $\sum_{i \notin S} a_i > A$. Then the integrality of the numbers a_i yields $\sum_{i \notin S} w_i = \sum_{i \notin S} a_i \geq A + 1$. The jobs in $\{1, 2, \dots, n\} \setminus S$ are all processed after job $n + 1$, and hence have completion times of at least $p_{n+1} = M$. Then, the total cost of the solution is at least $M \sum_{i \notin S} w_i \geq M(A + 1) > MA + 4A^2$, where the last inequality follows because $M > 4A^2$. But now the objective value of π would be greater than $4A^2 + MA$, the desired contradiction. Therefore, $\sum_{i \notin S} a_i \leq A$ holds, which together with (2) yields $\sum_{i \in S} a_i = A$. Hence, the PARTITION instance has answer YES.

Summarizing, this yields the following theorem.

Theorem 1. *Problem CWSCT is NP-hard in the ordinary sense.*

3 The Polynomial Time Approximation Scheme

Let $\varepsilon > 0$ be the desired precision of approximation (a fixed constant that is not part of the input); to simplify the presentation we will assume that $1/\varepsilon$ is integer.

Consider an instance I of CWSCT. Let $W = \sum_{j=1}^n w_j$ denote the total w -weight of the jobs, and let $P = \sum_{j=1}^n p_j$ denote their total processing time. We assume without loss of generality that instance I has at least one feasible schedule in (1); this can be checked in polynomial time through Smith's algorithm [10]. Finally, we fix an optimal schedule π^* for instance I relative to which we will do our analysis.

The approximation scheme goes through a number of *guessing* steps that guess certain pieces of information about the structure of the optimal schedule. All our guesses can be emulated in polynomial time either by means of a binary search or by means of total enumeration.

In our first guessing step, we guess the optimal objective value OPT . Since OPT lies somewhere between 0 and WP , we can approximate it with arbitrary precision through a binary search: If our current guess is too small, the whole procedure will fail, and we know that we have to use a larger guess. If our current guess is too large, the procedure yields an approximate schedule and we may move on to a smaller guess. Hence, the guesses eventually converge to OPT , and we terminate the binary search as soon as the induced lower and upper bounds on OPT are only a factor $1 + \varepsilon$ away from each other.

Based on OPT , we define the threshold parameter τ :

$$\tau = \varepsilon \text{OPT}. \quad (3)$$

A pair of jobs (i, j) is a *critical* pair if $w_i p_j - w_j p_i \geq \tau$ holds. A critical pair (i, j) is called an *active* critical pair in π if $\pi_j < \pi_i$. A critical pair (i, j) is called an *in-active* critical pair in π if $\pi_j > \pi_i$.

Lemma 1. *In any optimal schedule, there are at most $1/\varepsilon$ active critical pairs.*

Proof. Consider an optimal schedule, and assume that some job i participates in exactly k active critical pairs $(i, j_1), (i, j_2), \dots, (i, j_k)$. Then the contribution of job i to the objective value is at least

$$w_i C_i \geq w_i \sum_{l=1}^k p_{j_l} \geq \sum_{l=1}^k (w_i p_{j_l} - w_{j_l} p_i) \geq k \varepsilon \text{OPT}.$$

Therefore, there are at most $1/\varepsilon$ active critical pairs. \square

As an immediate consequence of Lemma 1, any optimal schedule has at most $2/\varepsilon$ jobs that participate in some active critical pair. In our second guessing step, we guess these at most $2/\varepsilon$ jobs together with their relative ordering in the optimal schedule π^* . This guessing step amounts to enumerate $O(n^{2/\varepsilon})$ subsets and $(2/\varepsilon)!$ total orderings for each of these subsets, that is, to check a polynomial number of possibilities. The guessed ordering of the guessed jobs yields a chain-like precedence constraint structure **Chain** on the job set.

Now we introduce our first version of the Lagrangian relaxation of (1). For a real parameter $\lambda \geq 0$, we define Lagrangian job weights $\ell_j^\lambda = w_j + \lambda u_j$ where $1 \leq j \leq n$. Consider the following optimization problem $P(\lambda)$:

$$\begin{aligned} P(\lambda) = \min \sum_{j=1}^n \ell_j^\lambda C_j^\pi - \lambda U &= \sum_{j=1}^n w_j C_j^\pi + \lambda (\sum_{j=1}^n u_j C_j^\pi - U) \\ \text{s.t. } \pi \text{ satisfies } \text{Chain} \end{aligned} \quad (4)$$

For every $\lambda \geq 0$ the value $P(\lambda)$ is a lower bound on OPT , since the optimal permutation π^* for (1) by definition satisfies the precedence constraints **Chain** and also the budget constraint $\sum_j u_j C_j \leq U$.

Moreover, for every $\lambda \geq 0$, problem $P(\lambda)$ can be solved in polynomial time: It is the scheduling problem of minimizing total weighted job completion time on a single machine under a single chain precedence constraint. In Section 4 we explain how Horn's algorithm [4] solves this problem by grouping the jobs in the chain **Chain** into a number of bundles. Note that some of these bundles may consist of only one job. For every λ with weights ℓ_j^λ , we denote the resulting set of bundles by $B_\lambda = \{J_1, J_2, \dots\}$. For every bundle J , we denote by p_J the total processing time of all the jobs in J and by w_J their total w -weight. We define the set D_λ of *dangerous* jobs as:

$$D_\lambda = \{i \notin B_\lambda : \exists J \in B_\lambda \text{ with } |p_J w_i - p_i w_J| \geq \tau\}. \quad (5)$$

Note that a dangerous job is never a job bundle, and also is not part of a bundle. A dangerous job $i \in D_\lambda$ becomes a *misplaced* dangerous job, if there exists a job bundle $J \in B_\lambda$ such that either $w_i p_J - w_J p_i \geq \tau$ and i is scheduled before J in the optimal schedule π^* , or $w_J p_i - w_i p_J \geq \tau$ and J is scheduled before i in the optimal schedule π^* .

Lemma 2. *With respect to any real $\lambda \geq 0$, there are at most $1/\varepsilon$ misplaced dangerous jobs in the optimal schedule π^* .*

Proof. Assume that job i is a misplaced dangerous job with respect to $J_1, J_2, \dots, J_k \in B_\lambda$, and that i is scheduled after all these jobs J_j . Then

$$w_i C_i \geq \sum_{j=1}^k w_i p_{J_j} \geq \sum_{j=1}^k (w_i p_{J_j} - p_{J_j} w_i) \geq k \tau \geq k \varepsilon \text{OPT}.$$

For $J \in B_\lambda$, let i_1, i_2, \dots, i_k be the set of misplaced dangerous jobs with respect to J , such that i_l is scheduled before J for all indices l . Then,

$$\begin{aligned} \sum_{j \in J} w_j C_j &\geq \sum_{j \in J} \sum_{l=1}^k w_j p_{i_l} = \sum_{l=1}^k w_J p_{i_l} \geq \sum_{l=1}^k (w_J p_{i_l} - p_J w_{i_l}) \\ &\geq k \tau \geq k \varepsilon \text{OPT}. \end{aligned}$$

Therefore, there are at most $1/\varepsilon$ misplaced dangerous jobs. \square

Our next goal is to find a maximizer λ^* for (4), to find a corresponding schedule and corresponding bundles, and to be in full control of the resulting misplaced dangerous jobs. This is accomplished through our third guessing step: We guess the set of misplaced dangerous jobs, and for each such misplaced dangerous job we guess its exact position within **Chain**. The resulting new precedence constraints are denoted by **Chain'**. Note that this guessing step amounts to enumerate $O(n^{1/\varepsilon})$ subsets and $O((2/\varepsilon)^{1/\varepsilon})$ possibilities for the positions; all in all that is a polynomial amount of work. We arrive at our second version $P'(\lambda)$ of the Lagrangian relaxation:

$$\begin{aligned} P'(\lambda) = \min & \sum_{j=1}^n \ell_j^\lambda C_j^\pi - \lambda U = \sum_{j=1}^n w_j C_j^\pi + \lambda (\sum_{j=1}^n u_j C_j^\pi - U) \\ \text{s.t. } & \pi \text{ satisfies } \textbf{Chain}' \end{aligned} \quad (6)$$

For the rest of this section, we will keep the old notation of job bundles exactly as introduced for $P(\lambda)$, and we will not adapt it to the second relaxation $P'(\lambda)$. Note that for all $\lambda \geq 0$ the value $P'(\lambda)$ is (again) a lower bound on OPT , since the optimal permutation π^* satisfies the precedence constraints **Chain'** and the budget constraint $\sum_j u_j C_j \leq U$.

Below we show how to determine a maximizer λ^* for (6) in polynomial time. If the corresponding bundles and misplaced difficult jobs collide with the data of our third guessing step, we simply cut off this guessing branch and move on to the next guess.

Lemma 3. *We may assume without loss of generality that for $\lambda = 0$ the corresponding optimal solution to $P'(\lambda)$ does not satisfy the budget constraint $\sum_{j=1}^n u_j C_j \leq U$.*

For $\lambda = 2WP$ the corresponding optimal solution to $P'(\lambda)$ does satisfy the budget constraint $\sum_{j=1}^n u_j C_j \leq U$.

Proof. First we assume $\lambda = 0$. Then, the objective function in (6) becomes $\sum_{j=1}^n w_j C_j$, that is, it becomes the objective function of (1). Since furthermore the optimal permutation π^* for (1) is feasible for (6), we have $P'(0) \leq \text{OPT}$. In case the optimal schedule for (6) satisfies the budget constraint in (1), it is feasible for (1) and hence optimal for (1). In this case there is nothing left to do.

Next we assume $\lambda = 2WP$. If a solution π for (6) does not satisfy the budget constraint, then $\sum_{j=1}^n u_j C_j^\pi \geq U+1$ and its objective value is at least $\lambda = 2WP$. If a solution π for (6) satisfies the budget constraint (as for instance the solution π^* does), then its objective value is at most WP . Hence, in this case the optimal solution for (6) satisfies the budget constraint. \square

For technical reasons, we now replace the budget bound U by the infinitesimally larger bound $U+\gamma$. This does not change the combinatorics of the problem, since all the job weights, job processing times, and job completion times are integers. However, it helps us to get rid of certain degeneracies. Our next step is to determine in polynomial time a maximizer λ^* for the second Lagrangian relaxation $P'(\lambda^*)$. This can be done by a straightforward binary search that is based on the following four observations:

- The cost of every permutation π as a function of λ is a linear function. This function crosses the line $\lambda = 0$ in some integer between 0 and WP , and crosses the line $\lambda = 2WP$ in some integer between $-2WPU$ and $2WP^2 \sum_j u_j$. Since we are using the modified budget bound $U + \gamma$, and since all the job data are integers, this linear function can not have slope zero.
- The cost of the optimal solution of (6) as a function of λ is the minimum of a set of linear functions (one linear function for each permutation). Therefore, this cost is a piecewise linear, concave function in λ .
- The (absolute value of the) difference between the slopes of two non-parallel linear functions is at least $1/(2WP)$ and at most $2WP$.
- The distance between two adjacent break-points on the lower envelope of these functions is at least $1/(2WP)$.

Together with the maximizer λ^* we also compute two optimal solutions α and β for $P'(\lambda^*)$, such that the first solution α satisfies the budget constraint $\sum_{j=1}^n u_j C_j^\alpha \leq U$, whereas the second solution β violates it with $\sum_{j=1}^n u_j C_j^\beta > U$. The existence of two such solutions can be established by standard observations: The optimal permutations for $\lambda = \lambda^* - \delta$ and $\lambda = \lambda^* + \delta$ for some arbitrarily small $\delta > 0$ are also optimal for λ^* . If all optimal solutions for λ^* would violate the budget constraint, then $P'(\lambda^* + \delta) > P'(\lambda^*)$ and λ^* would not maximize (6). Similarly, if all optimal solutions for λ^* would satisfy the budget constraint, then they would satisfy it with strict inequality (that's a consequence of the modified budget bound $U + \gamma$). But then $P'(\lambda^* - \delta) > P'(\lambda^*)$, and again λ^* would not maximize (6).

In general, when λ is changing then the corresponding set of job bundles in the corresponding optimal schedule is also changing. By using small perturbations of the job processing times, we may assume that in the point λ^* at most one of the following events happens:

- Two pieces J' and J'' (that may be job bundles or single jobs) are merged into a single job bundle J .
- A job bundle J is split into two pieces J' and J'' (that again may be job bundles or simple jobs).

Now we consider a minimal sequence of transpositions that transforms permutation α into β . In each such transposition some job is swapped with another job. Since α and β both are optimal permutations for $P'(\lambda^*)$, the swapped jobs must have the same Smith ratio $\ell_j^{\lambda^*}/p_j$. We now argue that such a transposition can not change the objective function a lot.

Lemma 4. *Every transposition increases $\sum_j w_j C_j$ by at most $2\varepsilon \text{OPT}$.*

Proof. At least one of the two swapped jobs is not a job bundle (since the ordering of job bundles is fully determined by the precedence constraints **Chain**).

First we consider the case where both swapped jobs i and i' are single jobs (and not job bundles). Without loss of generality we assume that in schedule α

job i is scheduled before job i' . Then, the swap increases $\sum_j w_j p_j$ by an amount of $w_i p_{i'} - p_i w_{i'}$. Since (i, i') is not an active critical pair, this increase is at most εOPT .

Next, consider the case where a single job i is swapped with a job bundle J . Since the schedules α and β place i and J in different relative orders, we conclude that job i can not be a dangerous job.

- If α runs job i before J , then the swap changes $\sum_j w_j C_j$ by $w_i p_J - p_i w_J$.
- If α runs job i after J , then the swap changes $\sum_j w_j C_j$ by $w_J p_i - p_J w_i$.

In any case, the objective value $\sum_j w_j C_j$ increases by at most $|w_i p_J - p_i w_J|$. We now prove that increase is bounded by $|w_i p_J - p_i w_J| \leq 2\varepsilon \text{OPT}$. We distinguish three subcases.

In the first subcase, we assume that the job bundle J does not change in λ^* . Since i is not a misplaced dangerous job, we read from (5) that $|w_i p_J - p_i w_J| \leq \tau \leq \varepsilon \text{OPT}$. This yields the desired inequality.

In the second subcase, we assume that the job bundle J is created in λ^* by merging two pieces J' and J'' (that were job bundles for some $\lambda = \lambda^* - \delta$ sufficiently close to λ^*). In this case we have

$$\begin{aligned} |w_i P_J - p_i w_J| &= |w_i P_{J'} - p_i w_{J'} + w_i P_{J''} - p_i w_{J''}| \\ &\leq |w_i P_{J'} - p_i w_{J'}| + |w_i P_{J''} - p_i w_{J''}| \\ &\leq 2\tau \\ &\leq 2\varepsilon \text{OPT}. \end{aligned}$$

In the third subcase, we assume that the job bundle J results from splitting a piece J' in λ^* (that is: for $\lambda < \lambda^*$, piece J' is a job bundle, and for some $\lambda = \lambda^* + \delta$ sufficiently close to λ^* , piece J' is replaced with two pieces J and J''). In this case we have

$$\begin{aligned} |w_i P_J - p_i w_J| &= |w_i(p_{J'} - p_{J''}) - p_i(w_{J'} - w_{J''})| \\ &= |(w_i p_{J'} - p_i w_{J'}) - (w_i p_{J''} - p_i w_{J''})| \\ &\leq |w_i p_{J'} - p_i w_{J'}| + |w_i p_{J''} - p_i w_{J''}| \\ &\leq 2\tau \\ &\leq 2\varepsilon \text{OPT}. \end{aligned}$$

Since we have settled all possible cases and subcases, the proof is complete. \square

Along the minimal sequence of transpositions that transforms permutation α into β , we return the first permutation that satisfies the budget constraint $\sum_j u_j C_j \leq U$. Every permutation in this sequence constitutes an optimal solution with respect to the job weights ℓ^{λ^*} . The permutation that comes just before the returned permutation satisfies $\sum_j w_j C_j \leq \text{OPT}$. By Lemma 4, the swapping of jobs leading to the next permutation increases the w -cost by at most $2\varepsilon \text{OPT}$. Hence, the returned solution has an objective value of at most $(1 + 2\varepsilon) \text{OPT}$. (And if we take into account that because of our first guessing step we are not working

with the exact value of OPT , but only with an approximation thereof, then this bound becomes the slightly weaker bound $(1 + 2\varepsilon)(1 + \varepsilon)\text{OPT}$. In any case, we have established the following result.

Theorem 2. *Problem CWSCT has a polynomial time approximation scheme.*

4 Something about Smith and Horn

In a classical paper in scheduling theory, Smith [10] describes a polynomial time algorithm for the problem of minimizing the sum of weighted job completion times on a single machine (without any precedence constraints): A simple exchange argument shows that the jobs should be ordered by non-increasing Smith ratios w_j/p_j .

In another classical paper, Horn [4] gives a polynomial time solution for the problem of minimizing the sum of weighted job completion times on a single machine under *tree-like* precedence constraints. In this paper, we are only interested in the following highly restricted special case: There are two job families K_1, \dots, K_k and L_1, \dots, L_m . The jobs in the first family are totally ordered by the single chain $K_1 \rightarrow K_2 \rightarrow \dots \rightarrow K_k$. The jobs in the second family are all independent of each other, and independent of the jobs in the first family.

Suppose that there are two consecutive jobs $K_a \rightarrow K_b$ in the chain with $w_a/p_a < w_b/p_b$ (that is, they ordered against their Smith ratios). Then, we may assume that no other job L_c is scheduled between these two jobs: If $w_c/p_c < w_b/p_b$, then L_c is better swapped with K_b , and if $w_c/p_c > w_a/p_a$, then L_c is better swapped with K_a . (A similar swapping argument works, in case there are more than one jobs scheduled between K_a and K_b .) Hence, in any optimal schedule K_a and K_b are consecutive, and we may merge them into a single new job with processing time $p_a + p_b$ and weight $w_a + w_b$. Optimal schedules for the new instance translate immediately into optimal schedules for the old instance, and vice versa. Note that the optimal objective value of the new instance equals $w_a p_b$ plus the objective value of the old instance; however, this shift in the objective value does not concern the structure of optimal solutions.

Horn's [4] algorithm repeats this merging operation over and over again, as long as there are consecutive (possibly merged) jobs in the chain that are ordered against their Smith ratios. When this procedure terminates, the chain has been cut into a number of so-called *bundles* of merged jobs. Some bundles may consist of only one job. For every bundle J , we denote by p_J the total processing time of all the jobs in J and by w_J their total weight. Along the chain the bundles are ordered by non-increasing Smith ratios w_J/p_J . Thus the precedence constraints become non-restrictive and disappear. We are left with a number of *job bundles* that result from the chain, and with a number of original (non-bundle, non-chain) jobs.

References

1. J.A. ASLAM, A. RASALA, C. STEIN, AND N.E. YOUNG (1999). Improved bicriteria existence theorems for scheduling. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'1999)*, 846–847.
2. M.R. GAREY AND D.S. JOHNSON (1979). *Computers and Intractability*. Freeman, San-Francisco.
3. M.X. GOEMANS AND R. RAVI (1996). The constrained minimum spanning tree problem. *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory (SWAT'1996)*, LNCS 1097, Springer Verlag, 66–75.
4. W.A. HORN (1972). Single-machine job sequencing with tree-like precedence ordering and linear delay penalties. *SIAM Journal on Applied Mathematics* 23, 189–202.
5. J.-H. LIN AND J.S. VITTER (1992). ϵ -approximations with minimum packing violation. *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing (STOC'1992)*, 771–782.
6. S.T. MCCORMICK AND M.L. PINEDO (1995). Scheduling n independent jobs on m uniform machines with both flowtime and makespan objectives: a parametric analysis. *ORSA Journal on Computing* 7, 63–77.
7. A. RASALA, C. STEIN, E. TORNG, AND P. UTHAISOMBUT (2002). Existence theorems, lower bounds and algorithms for scheduling to meet two objectives. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'2002)*, 723–731.
8. R. RAVI (2002). Bicriteria spanning tree problems. *Proceedings of the 5th Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'2002)*, LNCS 2462, Springer Verlag, 3–4.
9. D.B. SHMOYS AND E. TARDOS (1993). An approximation algorithm for the generalized assignment problem. *Mathematical Programming* 62, 461–474.
10. W.E. SMITH (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly* 3, 59–66.
11. C. STEIN AND J. WEIN (1997). On the existence of schedules that are near-optimal for both makespan and total-weighted completion time. *Operations Research Letters* 21, 115–122.

Near-Optimum Global Routing with Coupling, Delay Bounds, and Power Consumption

Jens Vygen

Research Institute for Discrete Mathematics, University of Bonn
Lennéstraße 2, 53113 Bonn, Germany
vygen@or.uni-bonn.de

Abstract. We propose the first theoretical approach to global routing that takes coupling between adjacent wires, bounds on delays along critical paths, and overall capacitance (power consumption) into account. It consists of an efficient combinatorial fully polynomial approximation scheme to a fractional relaxation, followed by randomized rounding. The overall deviation from the optimum can be bounded. The model could also be used for routing traffic flows with congestion-dependent travel times.

1 Introduction

Because of the enormous size of VLSI routing instances, detailed routing is usually preceded by global routing, defining an area for each net to which the path search in detailed routing will be restricted.

But global routing is also needed for another reason: not every placement allows a feasible routing. Moreover, a feasible routing may be possible only with intolerable detours. It is important to detect infeasible routing instances, and – if possible – a certificate of infeasibility, faster than by actually trying to find a feasible routing.

In its simplest version, the global routing problem amounts to packing Steiner trees, connecting given terminal sets, in an undirected graph with edge capacities. For this problem, Albrecht [2001a,2001b] described an approach that is very successful in practice. It consists of solving a fractional relaxation by a combinatorial fully polynomial approximation scheme (extending earlier work on multicommodity flows) and then applying randomized rounding.

However, neither this nor any other approach (see Hu and Sapatnekar [2001] for a survey) optimizes the main design objectives: timing, power consumption, and manufacturing yield. Minimizing the total length of the Steiner trees is not important, but minimizing a weighted sum of capacitances is: this is equivalent to minimizing power consumption. Delays on critical paths also depend on the capacitances of their nets. Even when assuming constant wire width, however, the capacitance can no longer be assumed to be proportional to the length, because coupling between neighbouring wires plays an increasingly important role. Often small detours are better than the densest possible packing. Spreading the

wires also helps to improve the yield. Recent papers on global routing (Xu et al. [2003], Jing et al. [2003], Ho et al. [2003]) propose heuristics partially addressing these objectives, but none of them can obtain any performance guarantee.

We describe the first global routing algorithm, with a provable performance guarantee, that takes coupling, timing, and power consumption into account directly. The yield is also improved as a by-product. Our algorithm extends earlier work on multicommodity flows, fractional global routing, and randomized rounding. After describing the problem in section 2, our main result will be a fully polynomial approximation scheme for a fractional relaxation, presented in section 3. In section 4 we show how to make randomized rounding work. Finally, in section 5, we conclude with remarks on related problems in routing traffic flow and general min-max resource sharing problems.

2 Global Routing in VLSI Layout

For this paper it does not matter whether or not detailed routing is done grid-based, and whether or not we allow diagonal wires. In any case, the chip area with its (currently approximately 8) routing planes is partitioned into regions (in the simplest case by an axis-parallel grid). The regions will form the vertices of the global routing graph. Adjacent regions are joined by an edge, whose capacity indicates how many wires of unit width can join the two regions while observing all routing constraints. A new promising approach to computing accurate capacities fast is described by Müller [2002].

For each net, we consider the set of regions that contain at least one of its pins. If a pin has shapes intersecting more than one region, we can assign it to one of them heuristically (e.g. the one that is closest to the center of gravity of the net, or based on the solution to the corresponding GROUP STEINER TREE PROBLEM). If a net has all pins in one region, it is ignored in global routing. Usually it is wise to wire nets with very short connections beforehand and have them as blockages in the subsequent routing process. This allows more accurate capacity estimations. Of course, one may reroute such nets later if deemed necessary.

In the following, let G be the global routing graph, with edge capacities $u : E(G) \rightarrow \mathbb{R}$ and lengths $l : E(G) \rightarrow \mathbb{R}_+$. Let \mathcal{N} be the set of nets. For each $N \in \mathcal{N}$ we have a set \mathcal{Y}_N of feasible Steiner trees. For example, we may restrict attention to Elmore-delay-optimal Steiner trees for certain critical nets, or impose length bounds. For the following it is not even important to know what nets and Steiner trees are. The only assumption we make is that each Steiner tree Y is associated with an edge set $E(Y)$, and we can find for any $N \in \mathcal{N}$ and any $\psi : E(G) \rightarrow \mathbb{R}_+$ an element $Y \in \mathcal{Y}_N$ with $\sum_{e \in E(Y)} \psi(e)$ (almost) minimum sufficiently fast. In fact, in most cases \mathcal{Y}_N will simply contain all Steiner trees for N in G .

Moreover, we assume to have constants $w(N, e) \in \mathbb{R}_+$ describing the width of a wire for net N that uses edge e . For an integer programming formulation, we choose a Steiner tree for each net by binary variables $x_{N,Y}$ ($N \in \mathcal{N}$, $Y \in \mathcal{Y}_N$).

Then a traditional version of the global routing problem can be formulated as follows:

$$\begin{aligned}
 \min \quad & \sum_{N \in \mathcal{N}} c(N) \sum_{e \in E(G)} l(e) \sum_{Y \in \mathcal{Y}_N : e \in E(Y)} x_{N,Y} \\
 \text{s.t.} \quad & \sum_{N \in \mathcal{N}} \sum_{Y \in \mathcal{Y}_N : e \in E(Y)} w(N, e) x_{N,Y} \leq u(e) \quad \text{for } e \in E(G) \\
 & \sum_{Y \in \mathcal{Y}_N} x_{N,Y} = 1 \quad \text{for } N \in \mathcal{N} \\
 & x_{N,Y} \in \{0, 1\} \quad \text{for } N \in \mathcal{N} \text{ and } Y \in \mathcal{Y}_N
 \end{aligned} \tag{1}$$

To decide whether an instance of this problem has a feasible solution is NP-complete even in the case of a planar grid with unit capacities and two-terminal nets only (Raghavan [1986]). This remains true if we impose length bounds, even if all nets must be routed shortest possible (Vygen [1994]).

Before extending this classical global routing model, let us make some comments on previous work. For a comprehensive survey on various global routing approaches we refer to Hu and Sapatnekar [2001].

The only approach for which a good theoretical performance guarantee can be proved (under reasonable assumptions) goes back to Raghavan and Thompson [1987, 1991], who proposed to solve an LP relaxation first, and then use randomized rounding to obtain an integral solution whose maximum capacity violation can be bounded. This also proves a bound on the integrality gap, as we shall see in Section 4. A similar approach with a more complicated rounding step has been proposed by Karp et al. [1987].

Together with polynomial-time linear programming algorithms, randomized rounding yields an approximation algorithm with respect to the objective of minimizing the maximum relative congestion

$$\max_{e \in E(G)} \frac{1}{u(e)} \sum_{N \in \mathcal{N} : e \in E(Y_N)} w(N, e). \tag{2}$$

However, for instances corresponding to current complex chips (with millions of edges and nets), all exact algorithms are too slow for solving even the LP relaxation optimally. Therefore fast approximation algorithms are interesting. Even better, there exist combinatorial fully polynomial approximation schemes, i.e. algorithms that compute a feasible (fractional) solution whose LP value is within a factor of $1 + \epsilon$ of the optimum, and whose running time is bounded by a polynomial in $|V(G)|$ and $\frac{1}{\epsilon}$ (and is less than needed for general linear programming). Most research has focused on the special case where $w \equiv 1$, every net has exactly two elements, and \mathcal{Y}_N contains all paths connecting N . This is the EDGE-DISJOINT PATHS PROBLEM, and its fractional relaxation is known as the MULTICOMMODITY FLOW PROBLEM. The first combinatorial fully polynomial approximation scheme for multicommodity flows was developed by Shahrokhi and Matula [1990]; see Karakostas [2002] for the latest improvement and more references.

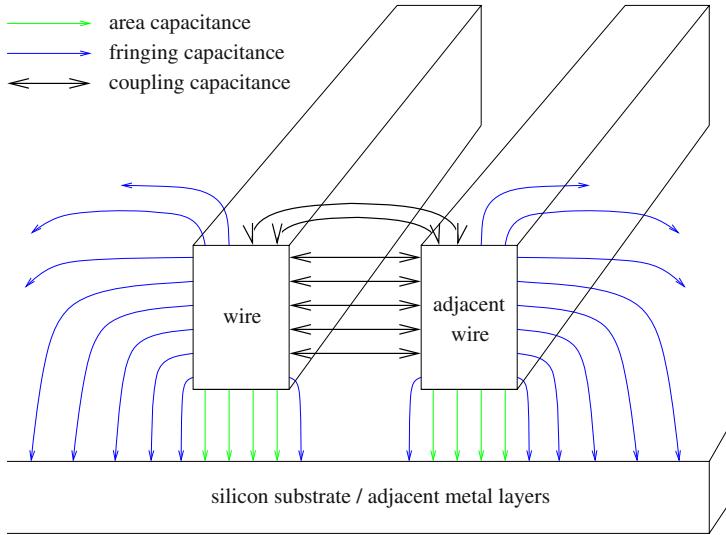


Fig. 1.

Approximation algorithms for multicommodity flow problems have been applied to global routing first by Carden, Li and Cheng [1996]. Later, Albrecht [2001a,2001b] applied one of the latest approximation algorithms, due to Garg and Könemann [1998], to the LP relaxation of (1), and combined it with ideas of several others in order to obtain a better running time in practice (without increasing the theoretical worst-case running time). We will now generalize this approach to take the important design objectives into account.

The power consumption of a chip – as far as influenced by routing – is proportional to the weighted sum of all capacitances, weighted by switching activities. The capacitance of a net consists of area capacitance (proportional to length times width), fringing capacitance (proportional to length) and coupling capacitance (proportional to the length if adjacent wires exist); see Figure 1. Of course, the coupling capacitance also depends on the distance of the two parallel wires. In older technologies, the impact of coupling was quite small and often ignored. This is no longer feasible with currently used deep-submicron technologies.

We model the capacitance as follows. Let $l(e, N)$ be the maximum capacitance of a wire along global routing edge e , assuming parallel wires running at both sides at minimum distance. As we fix the width of each pair of net and edge in advance, these numbers are known. Now we assume that extra space assigned to a wire reduces the capacitance linearly, up to some maximum space. If we reserve space $w(N, e) + y(e, N)s(e, N)$, where $0 \leq y(e, N) \leq 1$, we have a capacitance $l(e, N) - y(e, N)v(e, N)$. This is a simplification in two respects: first,

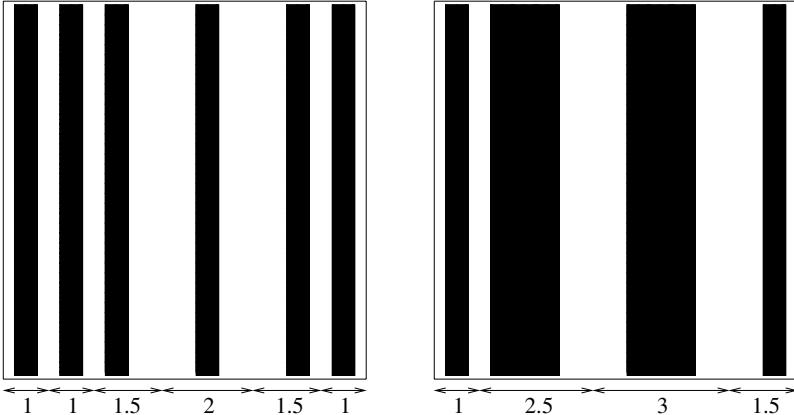


Fig. 2.

coupling capacitance does not depend linearly on the distance, and secondly, it may not always be possible to have all wires with $y(e, N) = 1$ next to each other.

Often we will have $v(e, N) = v(e, N')$ and $s(e, N) = s(e, N')$ for any two nets N and N' . However, the more general model causes no additional difficulties. Figure 2 shows a channel corresponding to a global routing edge e of capacity $u(e) = 8$ with two sets of wires using it. In the first case (to the left) we have six wires of unit width, in the second case (to the right) we have two unit width wires and two double width wires. In each case, the wires use up 6 units of capacity. For maximum spreading (one extra unit free space between each pair of adjacent wires we would need capacity 12 in the first case and 10 in the second one. Assuming $s(e, N) = v(e, N) = 1$ for all nets N , the two layouts shown the nets (from left to right) get capacitance reductions 0, 0, $\frac{1}{2}$, 1, $\frac{1}{2}$, 0 and 0, $\frac{1}{2}$, 1, $\frac{1}{2}$. Note that in the presence of two wires with capacitance reduction 0 and 1 we must have at least two wires with coupling capacitance in between. In our model, this is ignored, and also blockages, pin shapes and vias are ignored. Nevertheless quite accurate results can be obtained.

Minimizing power consumption rather than total netlength will automatically lead to a better spreading of the wires, and thus to increased yield. In addition, we can also take timing constraints into account. On the one hand, the set \mathcal{Y}_N of feasible Steiner trees may exclude those that lead to intolerable detours. On the other hand, we can impose upper bounds on weighted sums of capacitances for sets of nets. This can be applied, for example, to the set of nets in each critical path. One possible methodology is to do a first static timing analysis based on the assumption that every net has some expected capacitance (only Steiner trees below this bound will be in the set \mathcal{Y}_N), enumerate all paths that have negative slack under this assumption, compute the sensitivity of their nets to capacitance changes, and use these values to translate the delay bound to a bound on the weighted sum of capacitances for each path. To compute reasonable

expected capacitances one could apply weighted slack balancing (see Albrecht et al. [2002] for an efficient algorithm) using delay sensitivity and congestion information. The details will have to be developed by future research. We are very confident, however, that the following model will prove useful in practice.

GLOBAL ROUTING PROBLEM

- Instance:*
- An undirected graph G with capacities $u : E(G) \rightarrow \mathbb{R}_+$.
 - A set \mathcal{N} of so-called nets, and a family \mathcal{Y}_N of Steiner trees for each $N \in \mathcal{N}$. We assume that we can find a $Y \in \mathcal{Y}_N$ with minimum $\sum_{e \in E(Y)} \psi(e)$ fast for any $\psi : E(G) \rightarrow \mathbb{R}_+$.
 - Wire widths $w : \mathcal{N} \times E(G) \rightarrow \mathbb{R}_+$ and possible extra space $s : E(G) \times \mathcal{N} \rightarrow \mathbb{R}_+$.
 - Maximum capacitances $l : E(G) \times \mathcal{N} \rightarrow \mathbb{R}_+$, and coupling contributions $v : E(G) \times \mathcal{N} \rightarrow \mathbb{R}_+$.
 - A family \mathcal{M} of subsets of \mathcal{N} , where $\mathcal{N} \in \mathcal{M}$, bounds $U : \mathcal{M} \rightarrow \mathbb{R}_+$ and weights $c(M, N) \in \mathbb{R}_+$ for $N \in M \in \mathcal{M}$.

- Task:* Find a $Y_N \in \mathcal{Y}_N$ and numbers $0 \leq y(e, N) \leq 1$ for $e \in Y_N$, for each net $N \in \mathcal{N}$, such that

- $\sum_{N \in \mathcal{N}: e \in E(Y_N)} (w(N, e) + s(e, N)y(e, N)) \leq u(e)$ for each edge e ,
- $\sum_{N \in M} c(M, N) \sum_{e \in E(Y_N)} (l(e, N) - v(e, N)y(e, N)) \leq U(M)$ for $M \in \mathcal{M}$,
- minimizing $\sum_{N \in \mathcal{N}} c(\mathcal{N}, N) \sum_{e \in E(Y_N)} (l(e, N) - v(e, N)y(e, N))$.

The assumption that we can find optimum Steiner trees fast is justified as almost all nets in a reasonable practical instance have a small number of pins (less than 10). Note that a dynamic programming algorithm exists that runs in $O(3^k n + 2^k(m + n \log n))$ time, where k is the number of terminals, $n = |V(G)|$ and $m = |E(G)|$ (Erickson, Monma and Veinott [1987]).

3 Fractional Global Routing

First observe that it suffices to compute a feasible solution as we can impose an arbitrary upper bound $U(\mathcal{N})$ on

$$\sum_{N \in \mathcal{N}} c(\mathcal{N}, N) \sum_{e \in E(Y_N)} (l(e, N) + v(e, N)y(e, N)),$$

and apply binary search to find the optimum value of $U(\mathcal{N})$. Then we look for a feasible solution (λ, x, y) to the following LP, where x is integral and $\lambda = 1$. As

this is hard, we first solve the following LP approximately and then (in Section 4) apply randomized rounding.

$$\min \lambda$$

$$\begin{aligned} \text{s.t. } & \sum_{Y \in \mathcal{Y}_N} x_{N,Y} = 1 && \text{for } N \in \mathcal{N} \\ & \sum_{N \in M} c(M, N) \left(\sum_{Y \in \mathcal{Y}_N} \sum_{e \in E(Y)} l(e, N) x_{N,Y} - \sum_{e \in E(G)} v(e, N) y_{e,N} \right) \leq \lambda U(M) && \text{for } M \in \mathcal{M} \\ & \sum_{N \in \mathcal{N}} \left(\sum_{Y \in \mathcal{Y}_N : e \in E(Y)} w(N, e) x_{N,Y} + s(e, N) y_{e,N} \right) \leq \lambda u(e) && \text{for } e \in E(G) \\ & y_{e,N} \leq \sum_{Y \in \mathcal{Y}_N : e \in E(Y)} x_{N,Y} && \text{for } e \in E(G), N \in \mathcal{N} \\ & y_{e,N} \geq 0 && \text{for } e \in E(G), N \in \mathcal{N} \\ & x_{N,Y} \geq 0 && \text{for } N \in \mathcal{N}, Y \in \mathcal{Y}_N \end{aligned} \tag{3}$$

The corresponding dual LP is:

$$\begin{aligned} \max & \sum_{N \in \mathcal{N}} z_N \\ \text{s.t. } & \sum_{e \in E(G)} u(e) \omega_e + \sum_{M \in \mathcal{M}} U(M) \mu_M = 1 \\ & z_N \leq \sum_{e \in E(Y)} \left(l(e, N) \sum_{M \in \mathcal{M} : N \in M} c(M, N) \mu_M + w(N, e) \omega_e - \chi_{N,e} \right) && \text{for } N \in \mathcal{N}, Y \in \mathcal{Y}_N \\ & \chi_{e,N} \geq v(e, N) \sum_{M \in \mathcal{M} : N \in M} c(M, N) \mu_M - s(e, N) \omega_e && \text{for } e \in E(G), N \in \mathcal{N} \\ & \chi_{e,N} \geq 0 && \text{for } e \in E(G), N \in \mathcal{N} \\ & \omega_e \geq 0 && \text{for } e \in E(G) \\ & \mu_M \geq 0 && \text{for } M \in \mathcal{M} \end{aligned} \tag{4}$$

The dual LP enables us to compute lower bounds on the optimum LP value:

Lemma 1. *Let $\omega_e \in \mathbb{R}_+$ ($e \in E(G)$) and $\mu_M \in \mathbb{R}_+$ ($M \in \mathcal{M}$). Let $\psi_{N,e} :=$*

$$\min_{\delta \in \{0,1\}} \left((l(e, N) - \delta v(e, N)) \sum_{M \in \mathcal{M} : N \in M} c(M, N) \mu_M + (w(N, e) + \delta s(e, N)) \omega_e \right). \tag{5}$$

Then

$$\frac{\sum_{N \in \mathcal{N}} \min_{Y \in \mathcal{Y}_N} \sum_{e \in E(Y)} \psi_{N,e}}{\sum_{e \in E(G)} u(e)\omega_e + \sum_{M \in \mathcal{M}} U(M)\mu_M} \quad (6)$$

is a lower bound on the optimum LP value of (3) and (4).

Proof: Set $\chi_{e,N} := \max \{0, v(e, N) \sum_{M \in \mathcal{M}: N \in M} c(M, N) \mu_M - s(e, N) \omega_e\}$ and

$$z_N := \min_{Y \in \mathcal{Y}_N} \sum_{e \in E(Y)} \left(l(e, N) \sum_{M \in \mathcal{M}: N \in M} c(M, N) \mu_M + w(N, e) \omega_e - \chi_{N,e} \right). \quad (7)$$

After dividing all variables by $\sum_{e \in E(G)} u(e)\omega_e + \sum_{M \in \mathcal{M}} U(M)\mu_M$, we get a feasible solution (z, μ, ω, χ) of (4). \square

Now we can present our approximation algorithm. It uses four parameters $0 < \epsilon, \epsilon_1, \epsilon_2 \leq 1$, and $t \in \mathbb{N}$, which will be determined later to prove bounds on the approximation guarantee and running time. Moreover, we assume to have a lower bound \mathcal{L}_N on the length $\sum_{e \in E(Y)} (l(e, N) - v(e, N))$ of any Steiner tree $Y \in \mathcal{Y}_N$ ($N \in \mathcal{N}$). Often a nontrivial lower bound can be computed in constant time using distances of the terminals.

FRACTIONAL GLOBAL ROUTING ALGORITHM

Input: An instance of the GLOBAL ROUTING PROBLEM with $\mathcal{N} = \{1, \dots, k\}$.

Output: A feasible solution to (3).

- ① Set $\omega_e := \frac{1}{u(e)}$ for $e \in E(G)$ and $\mu_M := \frac{1}{U(M)}$ for $M \in \mathcal{M}$.
 Set $x_{i,Y} := 0$ for $i = 1, \dots, k$, $Y \in \mathcal{Y}_i$
 Set $y_{e,i} := 0$ for $e \in E(G)$ and $i = 1, \dots, k$.
 Set $Y_i := \emptyset$ for $i = 1, \dots, k$.

② For $p := 1$ to t do:

For $i := 1$ to k do:

Let $\psi_{i,e}$ be defined as in (5).

If $Y_i = \emptyset$ or $\sum_{e \in E(Y_i)} \psi_{i,e} > (1 + \epsilon_1)z_i$ then:

Let $Y_i \in \mathcal{Y}_i$ with $\sum_{e \in E(Y_i)} \psi_{i,e} \leq (1 + \epsilon_2) \min_{Y \in \mathcal{Y}_i} \sum_{e \in E(Y)} \psi_{i,e}$.

Set $z_i := \sum_{e \in E(Y_i)} \psi_{i,e}$.

Set $x_{i,Y_i} := x_{i,Y_i} + 1$.

For $e \in E(Y_i)$ do:

If $v(e, i) \sum_{M \in \mathcal{M}: i \in M} c(M, i) \mu_M < s(e, N) \omega_e$ then $\delta_e := 0$
else $\delta_e := 1$.

Set $y_{e,i} := y_{e,i} + \delta_e$.

Set $\omega_e := \omega_e e^{\frac{w(i,e) + \delta_e s(e,i)}{u(e)}}$.

For $M \in \mathcal{M}$ with $i \in M$ do:

Set $\mu_M := \mu_M e^{ec(M,i) \frac{l(e,i) - \delta_e v(e,i)}{U(M)}}$.

For $j \in M$ do:

Set $z_j := z_j + (1 + \epsilon_2) \mathcal{L}_j c(M, j) \mu_M \left(e^{ec(M,i) \frac{l(e,i) - \delta_e v(e,i)}{U(M)}} - 1 \right)$.

③ Set $x_{i,Y} := \frac{1}{t} x_{i,Y}$ for $i = 1, \dots, k$ and $Y \in \mathcal{Y}_i$.

Set $y_{e,i} := \frac{1}{t} y_{e,i}$ for $e \in E(G)$ and $i = 1, \dots, k$.

We can show that this algorithm computes an approximate solution to (3). The following proof (and the above algorithm) uses and extends ideas of Young [1995], Garg and Könemann [1998], Fleischer [2000], and Albrecht [2001a, 2001b].

Theorem 2. Let an instance of the GLOBAL ROUTING PROBLEM be given, and let $m = |E(G)|$. Let $\Lambda \geq 1$, $\Lambda \geq \frac{c(M,i) \sum_{e \in E(Y)} l(e,i)}{U(M)}$ for $i \in M \in \mathcal{M}$ and $Y \in \mathcal{Y}_i$, and $\Lambda \geq \frac{w_{i,e} + s(e,i)}{u(e)}$ for $i \in \mathcal{N}$ and $e \in E(G)$.

Let $\epsilon_1, \epsilon_2 \geq 0$ and $0 < \epsilon < 1$. Let $\epsilon' := \epsilon(1 + \frac{3}{4}\Lambda\epsilon)(1 + \epsilon_1)(1 + \epsilon_2)$. Assume that $\epsilon' \lambda^* < 1$, where λ^* is the LP value of (3). Let $t \in \mathbb{N}$.

Let $\rho := \frac{\epsilon'}{\epsilon(1 - \epsilon' \lambda^*)(1 - \frac{\log(m + |\mathcal{M}|)}{\epsilon t \lambda^*})}$.

Then the above algorithm with parameters $\epsilon, \epsilon_1, \epsilon_2, t$ computes a ρ -approximate solution to (3). More precisely, the vectors x and y computed by the algorithm and $\rho \lambda^*$ constitute a feasible solution to (3).

Proof: Let $\mu_M^{(p,j)}, \omega_e^{(p,j)}, Y_j^{(p,j)}, \delta_e^{(p,j)}$ be the values of the variables $\mu_M, \omega_e, Y_j, \delta_e$ at the end of the j -th inner iteration within the p -th outer iteration of the algorithm ($e \in E(G), M \in \mathcal{M}$), and let $\psi_{i,e}^{(p,j)}$ be defined as in (5) with respect to $\mu_M^{(p,j)}$ and $\omega_e^{(p,j)}$ ($e \in E(G), i, j = 1, \dots, k, M \in \mathcal{M}$). (So in iteration (p, j) we use $\psi_{j,e}^{(p,j-1)}$ as edge lengths.) Let $\mu_M^{(p)} := \mu_M^{(p,k)}$ and $\omega_e^{(p)} := \omega_e^{(p,k)}$ be the values

at the end of iteration p . We write $\mu_M^{(p,0)} := \mu_M^{(p-1)}$ etc., and $\mu_M^{(0)}$ is the value of μ_M after ①.

By construction we have $0 \leq y_{e,i} \leq \sum_{Y \in \mathcal{Y}_i : e \in E(Y)} x_{i,Y}$ for $e \in E(G)$, $i \in \{1, \dots, k\}$, and $x_{i,Y} \geq 0$ for $i \in \{1, \dots, k\}$, $Y \in \mathcal{Y}_i$. We will check for which value of λ the other constraints of (3) hold.

Initially (after ①) we have

$$\sum_{e \in E(G)} u(e)\omega_e^{(0)} + \sum_{M \in \mathcal{M}} U(M)\mu_M^{(0)} = m + |\mathcal{M}|. \quad (8)$$

We first estimate the increase of the y -variables. We have

$$\begin{aligned} & \sum_{M \in \mathcal{M}} U(M)\mu_M^{(p,i)} \\ &= \sum_{M \in \mathcal{M} : i \notin M} U(M)\mu_M^{(p,i-1)} + \sum_{M \in \mathcal{M} : i \in M} U(M)\mu_M^{(p,i-1)} e^{\epsilon c(M,i) \sum_{e \in E(Y_i)} \frac{l(e,i) - \delta_e^{(p,i)} v(e,i)}{U(M)}} \\ &\leq \sum_{M \in \mathcal{M}} U(M)\mu_M^{(p,i-1)} \\ &\quad + \epsilon(1 + \frac{3}{4}\Lambda\epsilon) \sum_{M \in \mathcal{M} : i \in M} \mu_M^{(p,i-1)} c(M,i) \sum_{e \in E(Y_i)} (l(e,i) - \delta_e^{(p,i)} v(e,i)), \end{aligned} \quad (9)$$

because $\epsilon \leq 1$, $c(M,i) \sum_{e \in E(Y_i)} (l(e,i) - \delta_e^{(p,i)} v(e,i)) \leq \Lambda U(M)$ for $i \in M \in \mathcal{M}$, and $e^x \leq 1 + x + \frac{3}{4}x^2$ for $0 \leq x \leq 1$.

Similarly,

$$\begin{aligned} & \sum_{e \in E(G)} u(e)\omega_e^{(p,i)} \\ &= \sum_{e \notin E(Y_i)} u(e)\omega_e^{(p,i-1)} + \sum_{e \in E(Y_i)} u(e)\omega_e^{(p,i-1)} e^{\epsilon \frac{w(i,e) + \delta_e^{(p,i)} s(e,i)}{u(e)}} \\ &\leq \sum_{e \in E(G)} u(e)\omega_e^{(p,i-1)} + \epsilon(1 + \frac{3}{4}\Lambda\epsilon) \sum_{e \in E(Y_i)} (w(i,e) + \delta_e^{(p,i)} s(e,i))\omega_e^{(p,i-1)}. \end{aligned} \quad (10)$$

Combining (9) and (10) we get

$$\begin{aligned} & \sum_{e \in E(G)} u(e)\omega_e^{(p)} + \sum_{M \in \mathcal{M}} U(M)\mu_M^{(p)} \\ &\leq \sum_{e \in E(G)} u(e)\omega_e^{(p-1)} + \sum_{M \in \mathcal{M}} U(M)\mu_M^{(p-1)} + \epsilon(1 + \frac{3}{4}\Lambda\epsilon) \sum_{i=1}^k \sum_{e \in E(Y_i)} \psi_{i,e}^{(p,i-1)}. \end{aligned} \quad (11)$$

At any stage we have $z_i \leq (1 + \epsilon_2) \min_{Y \in \mathcal{Y}_i} \sum_{e \in E(Y)} \psi_{i,e}$: the ψ -variables never decrease, and if the z -variables increase (say by $(1 + \epsilon_2)\alpha$), then $\sum_{e \in E(Y)} \psi_{i,e}$ increases by at least α for any $Y \in \mathcal{Y}_i$. We conclude:

$$\sum_{e \in E(Y_i)} \psi_{i,e}^{(p,i-1)} \leq (1 + \epsilon_1)z_i \leq (1 + \epsilon_1)(1 + \epsilon_2) \min_{Y \in \mathcal{Y}_i} \sum_{e \in E(Y)} \psi_{i,e}^{(p)}. \quad (12)$$

Hence, combining (11) and (12),

$$\begin{aligned} & \sum_{e \in E(G)} u(e)\omega_e^{(p)} + \sum_{M \in \mathcal{M}} U(M)\mu_M^{(p)} \\ & \leq \sum_{e \in E(G)} u(e)\omega_e^{(p-1)} + \sum_{M \in \mathcal{M}} U(M)\mu_M^{(p-1)} + \epsilon' \sum_{i=1}^k \min_{Y \in \mathcal{Y}_i} \sum_{e \in E(Y)} \psi_{i,e}^{(p)}, \end{aligned} \quad (13)$$

where $\epsilon' := \epsilon(1 + \frac{3}{4}\Lambda\epsilon)(1 + \epsilon_1)(1 + \epsilon_2)$.

By Lemma 1 and weak linear programming duality,

$$\lambda_{lb} := \frac{\sum_{i=1}^k \min_{Y \in \mathcal{Y}_i} \sum_{e \in E(Y)} \psi_{i,e}^{(p)}}{\sum_{e \in E(G)} u(e)\omega_e^{(p)} + \sum_{M \in \mathcal{M}} U(M)\mu_M^{(p)}} \quad (14)$$

is a lower bound on the optimum LP value λ^* . Assuming $\epsilon'\lambda^* < 1$, (13) yields

$$\begin{aligned} & \sum_{e \in E(G)} u(e)\omega_e^{(p)} + \sum_{M \in \mathcal{M}} U(M)\mu_M^{(p)} \\ & \leq \frac{1}{1 - \epsilon'\lambda_{lb}^{(p)}} \left(\sum_{e \in E(G)} u(e)\omega_e^{(p-1)} + \sum_{M \in \mathcal{M}} U(M)\mu_M^{(p-1)} \right). \end{aligned} \quad (15)$$

Combining (8) and (15) and setting $\lambda_{lb} := \max_{p=1}^t \lambda_{lb}^{(p)}$, we get

$$\begin{aligned} \sum_{e \in E(G)} u(e)\omega_e^{(p)} + \sum_{M \in \mathcal{M}} U(M)\mu_M^{(p)} & \leq \frac{m + |\mathcal{M}|}{(1 - \epsilon'\lambda_{lb})^p} \\ & = (m + |\mathcal{M}|) \left(1 + \frac{\epsilon'\lambda_{lb}}{1 - \epsilon'\lambda_{lb}} \right)^p \\ & \leq (m + |\mathcal{M}|) e^{\frac{\epsilon'\lambda_{lb} p}{1 - \epsilon'\lambda_{lb}}}, \\ & \leq (m + |\mathcal{M}|) e^{\frac{\epsilon'\lambda_{lb} p}{1 - \epsilon'\lambda^*}}, \end{aligned} \quad (16)$$

as $1 + x \leq e^x$ for $x \geq 0$. For $p = t$ we get

$$\lambda_{lb} \geq \frac{1 - \epsilon'\lambda^*}{\epsilon't} \ln \left(\frac{\mathcal{U}}{m + |\mathcal{M}|} \right), \quad (17)$$

where $\mathcal{U} := \sum_{e \in E(G)} u(e)\omega_e^{(t)} + \sum_{M \in \mathcal{M}} U(M)\mu_M^{(t)}$.

For an upper bound on the lengths, let $M \in \mathcal{M}$. At termination we have

$$\begin{aligned}
& \sum_{i \in M} c(M, i) \left(\sum_{Y \in \mathcal{Y}_i} \sum_{e \in E(Y)} l(e, i) x_{i,Y} - \sum_{e \in E(G)} v(e, i) y_{e,i} \right) \\
&= \frac{1}{t} \sum_{p=1}^t \sum_{i \in M} \sum_{e \in Y_i^{(p)}} c(M, i) (l(e, i) - \delta_e^{(p,i)} v(e, i)) \\
&= \frac{U(M)}{\epsilon t} \ln \left(\prod_{p=1}^t \prod_{i \in M} \prod_{e \in Y_i^{(p)}} e^{\epsilon c(M, i) \frac{l(e, i) - \delta_e^{(p,i)} v(e, i)}{U(M)}} \right) \\
&= \frac{U(M)}{\epsilon t} \ln \left(U(M) \mu_M^{(t)} \right) \\
&\leq \frac{U(M)}{\epsilon t} \ln \mathcal{U}
\end{aligned} \tag{18}$$

Finally, we have for $e \in E(G)$:

$$\begin{aligned}
& \sum_{i=1}^k \left(\sum_{Y \in \mathcal{Y}_i : e \in E(Y)} w(i, e) x_{i,Y} + s(e, i) y_{e,i} \right) \\
&= \frac{1}{t} \sum_{p=1}^t \sum_{i \in \{1, \dots, k\} : e \in Y_i^{(p)}} (w(i, e) + \delta_e^{(p,i)} s(e, i)) \\
&= \frac{u(e)}{\epsilon t} \ln \left(\prod_{p=1}^t \prod_{i \in \{1, \dots, k\} : e \in Y_i^{(p)}} e^{\frac{w(i, e) + \delta_e^{(p,i)} s(e, i)}{u(e)}} \right) \\
&= \frac{u(e)}{\epsilon t} \ln \left(u(e) \omega_e^{(t)} \right) \\
&\leq \frac{u(e)}{\epsilon t} \ln \mathcal{U}
\end{aligned} \tag{19}$$

By (18) and (19), we can choose $\lambda = \lambda_{ub} := \frac{1}{\epsilon t} \ln \mathcal{U}$ and have a feasible solution of (3). Comparing this upper bound with the lower bound (17), we can bound the approximation ratio ρ by

$$\rho \leq \frac{\lambda_{ub}}{\lambda_{lb}} \leq \frac{\epsilon' \ln \mathcal{U}}{\epsilon(1 - \epsilon' \lambda^*) \ln \left(\frac{\mathcal{U}}{m + |\mathcal{M}|} \right)}. \tag{20}$$

As $\mathcal{U} = e^{\epsilon t \lambda_{ub}} \geq e^{\epsilon t \lambda^*} = (m + |\mathcal{M}|)^\alpha$ with $\alpha = \frac{\epsilon t \lambda^*}{\log(m + |\mathcal{M}|)}$ we have $\frac{\ln \mathcal{U}}{\ln \left(\frac{\mathcal{U}}{m + |\mathcal{M}|} \right)} \leq \frac{\alpha}{\alpha - 1}$ and thus $\rho \leq \frac{\epsilon'}{\epsilon(1 - \epsilon' \lambda^*)} \cdot \frac{\alpha}{\alpha - 1}$. \square

Together with binary search, this theorem implies a fully polynomial approximation scheme: Assuming $\frac{1}{2} \leq \lambda^* \leq 1$ and $1 \leq \Lambda \leq 2$, and given $0 < \epsilon_0 < 1$, we

can choose ϵ_1 and ϵ_2 with $(1+\epsilon_1)(1+\epsilon_2) < \sqrt{1+\epsilon_0}$, $\epsilon := \frac{\epsilon_0}{27}$, and $t := \frac{\log(m+|\mathcal{M}|)}{\epsilon^2 \lambda^*}$ to obtain an approximation ratio of

$$\begin{aligned} \rho &= \frac{\epsilon'}{\epsilon(1-\epsilon'\lambda^*)(1-\epsilon)} \leq \sqrt{1+\epsilon_0}(1 + \frac{3}{4}\Lambda\epsilon) \left(\frac{1}{1-\epsilon'}\right)^2 \leq \sqrt{1+\epsilon_0} \left(\frac{1}{1-\frac{3}{2}\epsilon}\right)^3 \\ &\leq \sqrt{1+\epsilon_0} \left(\frac{1}{1-\frac{1}{18}\epsilon_0}\right)^3 \leq \sqrt{1+\epsilon_0} (1 + \frac{\epsilon_0}{9})^3 \leq 1 + \epsilon_0 \end{aligned} \quad (21)$$

in at most $\frac{1458 \ln(m+|\mathcal{M}|)}{\epsilon_0^2}$ iterations. Preliminary experience with practical instances, however, show much faster convergence and, with larger ϵ , good approximation guarantees already after 10–20 iterations.

Note that the FRACTIONAL GLOBAL ROUTING ALGORITHM also obtains an approximate dual solution, giving a certificate of infeasibility for most infeasible instances of the GLOBAL ROUTING PROBLEM. This is true because the integrality gap is quite small, as will be shown in the next section.

We close this section by remarking that the above approximation algorithm is quite similar to the oldest and simplest global routing strategy, often called rip-up and reroute. The idea is to relax the capacity constraints, route each net by a shortest Steiner tree and then reroute nets using overloaded edges. This can be regarded as a Lagrangian relaxation method. The main differences to the above algorithm are the update of the dual variables (= Lagrange multipliers) and the missing randomized rounding step: traditional rip-up and reroute algorithms just output the last Steiner tree found for each net. Rip-up and reroute algorithms are useful for very easy instances, and also to post-optimize a solution, e.g. to remove local capacity violations introduced by randomized rounding.

4 Randomized Rounding

In this section we show how to round a fractional solution to (3) without increasing λ too much. We need the following lemma by Raghavan and Spencer (see Raghavan [1988]), a variation of Chernoff's [1952] bound.

Lemma 3. *Let X_1, \dots, X_k be independent random variables in $[0, 1]$. Let μ be the sum of their expectations, and let $\epsilon > 0$. Then $X_1 + \dots + X_k \leq (1+\epsilon)\mu$ with probability at least $1 - e^{\mu(\epsilon-(1+\epsilon)\ln(1+\epsilon))}$.*

Proof: Let $\text{Prob}[\cdot]$ denote the probability of an event, and let $\text{Exp}[\cdot]$ denote the expectation of a random variable. Using $(1+\epsilon)^x \leq 1+\epsilon x$ for $0 \leq x \leq 1$ and $1+x \leq e^x$ for $x \geq 0$ we compute $\text{Prob}[X_1 + \dots + X_k \geq (1+\epsilon)\mu] = \text{Prob}\left[\frac{\prod_{i=1}^k (1+\epsilon)^{X_i}}{(1+\epsilon)^{(1+\epsilon)\mu}} \geq 1\right] = \text{Prob}\left[\frac{\prod_{i=1}^k (1+\epsilon X_i)}{(1+\epsilon)^{(1+\epsilon)\mu}} \geq 1\right] \leq \text{Exp}\left[\frac{\prod_{i=1}^k (1+\epsilon X_i)}{(1+\epsilon)^{(1+\epsilon)\mu}}\right] = \frac{\prod_{i=1}^k (1+\epsilon \text{Exp}[X_i])}{(1+\epsilon)^{(1+\epsilon)\mu}} \leq \frac{\prod_{i=1}^k e^{\epsilon \text{Exp}[X_i]}}{(1+\epsilon)^{(1+\epsilon)\mu}} = \frac{e^{\epsilon \mu}}{(1+\epsilon)^{(1+\epsilon)\mu}} = e^{(\epsilon-(1+\epsilon)\ln(1+\epsilon))\mu}.$ \square

Similarly as Raghavan [1988], we can now bound the effect of randomized rounding:

Theorem 4. Given a fractional solution (x, y, λ) to (3), we obtain a rounded solution $(\hat{x}, \hat{y}, \hat{\lambda})$ as follows: Independently for all $i \in \mathcal{N}$ we choose $Y \in \mathcal{Y}_i$ as Y_i with probability $x_{i,Y}$; then we set $\hat{x}_{i,Y_i} := 1$ and $\hat{x}_{i,Y} := 0$ for all $Y \in \mathcal{Y}_i \setminus \{Y_i\}$. Next we set $\hat{y}_{i,e} := \frac{y_{i,e}}{\sum_{Y \in \mathcal{Y}_i : e \in E(Y)} x_{i,Y}}$ if $e \in E(Y_i)$ and $\hat{y}_{i,e} := 0$ otherwise. Finally, we choose $\hat{\lambda}$ minimum possible such that $(\hat{x}, \hat{y}, \hat{\lambda})$ is a feasible solution to (3).

Let $\Lambda \leq \frac{U(M)}{c(M,i) \sum_{e \in E(Y)} l(e,i)}$ for $i \in M \in \mathcal{M}$ and $Y \in \mathcal{Y}_i$, and $\Lambda \leq \frac{u(e)}{w_{i,e} + s(e,i)}$ for $i \in \mathcal{N}$ and $e \in E(G)$.

Then $\hat{\lambda} \leq e^\alpha \lambda$ with probability at least $\frac{1}{2}$, where α is the solution to $1 + (\alpha - 1)e^\alpha = \frac{\ln(2(|\mathcal{M}| + |E(G)|))}{\lambda \Lambda}$.

Proof: We write $y'_{i,e} := \frac{y_{i,e}}{\sum_{Y \in \mathcal{Y}_i : e \in E(Y)} x_{i,Y}}$; then $\hat{y}_{i,e} = \sum_{Y \in \mathcal{Y}_i : e \in E(Y)} y'_{i,e} \hat{x}_{i,Y}$. $\hat{\lambda}$ is constrained by $|\mathcal{M}| + |E(G)|$ linear inequalities, which can be written as $A\hat{x} \leq \hat{\lambda}\mathbf{1}$, where $\mathbf{1}$ is an all-one vector, and the entries of the matrix A are $A_{M,i,Y} = \frac{c(M,i)}{U(M)} \sum_{e \in E(Y)} (l(e,i) - v(e,i)y'_{i,e})$ (for $i \in M \in \mathcal{M}$ and $Y \in \mathcal{Y}_i$) and $A_{e;i,Y} = \frac{1}{u(e)} (w(i,e) + s(e,i)y'_{i,e})$ (for $i \in \mathcal{N}$, $Y \in \mathcal{Y}_i$ and $e \in E(Y)$).

Note that

$$\Lambda \frac{c(M,i)}{U(M)} \sum_{Y \in \mathcal{Y}_i} \sum_{e \in E(Y)} (l(e,i) - v(e,i)y'_{i,e}) \hat{x}_{i,Y}, \quad (22)$$

$i = 1, \dots, k$, are independent random variables in $[0, 1]$. Similarly,

$$\Lambda \frac{1}{u(e)} (w(i,e) + s(e,i)y'_{i,e}) \sum_{Y \in \mathcal{Y}_i : e \in Y} \hat{x}_{i,Y}, \quad (23)$$

$i = 1, \dots, k$, are independent random variables in $[0, 1]$.

In each case, the sum of the expectations of the k random variables is at most $\lambda\Lambda$. Hence, by Lemma 3, their sum is at least $(1 + \epsilon)\lambda\Lambda$ with probability at most $e^{\lambda\Lambda(\epsilon - (1 + \epsilon)\ln(1 + \epsilon))}$. Setting $\epsilon = e^\alpha - 1$, this is $e^{-\lambda\Lambda(1 + (\alpha - 1)e^\alpha)} = \frac{1}{2(|\mathcal{M}| + |E(G)|)}$. \square

The algorithm can easily be derandomized by applying the method of conditional probabilities in a standard way (Raghavan [1988]), however, this may be too slow in practice. By allowing different values of Λ for the constraints, the above bound could be improved. Another attempt of improving the bound has been made by Leighton et al. [2001], who showed how the (de)randomized rounding can be improved by using an algorithmic version of the Lovász Local Lemma, exploiting the fact that most Steiner trees are short and their rounding does not affect regions that are far away.

In practice, violations introduced by randomized rounding are much smaller than guaranteed by the above theorem, and they can usually be removed by simple postprocessing heuristics.

5 Discussion

Note that the global routing problem could also be used to model optimizing traffic flows in a static setting (flow continuously repeated over time, like in rush-hour traffic), with hard capacities on edge flow rate and with transit times that depend piecewise linearly on the congestion. The algorithm is actually equivalent to selfish routing, but with respect to edge costs depending exponentially, rather than linearly, on the congestion. It is known that general selfish routing can lead to arbitrarily bad solutions (called user equilibria or Nash equilibria; cf. Roughgarden and Tardos [2000], Schulz and Stier [2003]). However, it is not clear whether this also holds for *fair* selfish routing, where the users (commodities, nets) take turns in finding alternate shortest routes (like in our algorithm). This may lead to better user equilibria. It is an interesting question whether there is a class of (reasonable) latency functions where fair selfish routing converges to the system optimum. Moreover, it is an open question whether the algorithm can be extended to a model in which coupling capacitances (travel times) depend non-linearly on the congestion.

Algorithms for packing linear programs, and more generally (nonlinear) min-max resource sharing problems, have been proposed before by many authors, including Grigoriadis and Khachiyan [1994,1996], Plotkin, Shmoys and Tardos [1995], Young [1995], and Garg and Könemann [1998]. However, the number of iterations in all these approaches depends at least linearly on the number of blocks (nets in our context) or on the number of resource constraints (edges plus capacitance constraints) or on the width (the value of the worst possible solution). In our algorithm the number of iterations is independent of the number of blocks and the width, and depends only logarithmically on the number of resource constraints. The dependency on the performance guarantee in all these algorithms is $O(\epsilon^{-2})$, like in our algorithm, or worse. It may be interesting to try to extend our approach to more general contexts.

With Dirk Müller, we are currently incorporating the presented algorithm into *BonnRoute*, an automatic routing tool used for designing many of the most complex industrial chips. We are confident that this will improve the quality of the chips (speed, power consumption), increase the manufacturing yield, and shorten turn-around times in design closure by avoiding semi-manual postprocessing.

References

- [2001a] Albrecht, C. [2001a]: Global routing by new approximation algorithms for multicommodity flow. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems* 20 (2001), 622–632
- [2001b] Albrecht, C. [2001b]: Zwei kombinatorische Optimierungsprobleme im VLSI-Design: Optimierung der Zykluszeit und der Slackverteilung und globale Verdrahtung. Ph.D. thesis, University of Bonn, 2001
- [2002] Albrecht, C., Korte, B., Schietke, J., and Vygen, J. [2002]: Maximum mean weight cycle in a digraph and minimizing cycle time of a logic chip. *Discrete Applied Mathematics* 123 (2002), 103–127.

- [1996] Carden IV, R.C., Li, J., and Cheng, C.-K. [1996]: A global router with a theoretical bound on the optimum solution. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15 (1996), 208–216
- [1952] Chernoff, H. [1952]: A measure of asymptotic efficiency for tests based on the sum of observations. *Annals of Mathematical Statistics* 23 (1952), 493–509
- [1987] Erickson, R.E., Monma, C.L., and Veinott jr., A.F. [1987]: Send-and-split method for minimum concave-cost network flows. *Mathematics of Operations Research* 12 (1987), 634–664
- [2000] Fleischer, L.K. [2000]: Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics* 13 (2000), 505–520
- [1998] Garg, N., and Könemann, J. [1998]: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science* (1998), 300–309
- [1994] Grigoriadis, M.D., and Khachiyan, L.G. [1994]: Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM Journal on Optimization* 4 (1994), 86–107
- [1996] Grigoriadis, M.D., and Khachiyan, L.G. [1996]: Coordination complexity of parallel price-directive decomposition. *Mathematics of Operations Research* 21 (1996), 321–340
- [2003] Ho, T.-Y., Chang, Y.-W., Chen, S.-J. and Lee, D.-T. [2003]: A fast crosstalk- and performance-driven multilevel routing system. *Proceedings of the IEEE International Conference on Computer-Aided Design* (Nov. 2003)
- [2001] Hu, J., and Sapatnekar, S.S. [2001]: A survey on multi-net global routing for interigated circuits. *Integration, the VLSI Journal* 31 (2001), 1–49
- [2003] Jing, T., Hong, X., Bao, H., Cai, Y., Xu, J., Cheng, C., Gu, J. [2003]: Utaco: a unified timing and congestion optimizing algorithm for standard cell global routing. *Proceedings of the Asia and South Pacific Design Automation Conference*, 2003, 834–839
- [2002] Karakostas, G. [2002]: Faster approximation schemes for fractional multicommodity flow problems. *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms* (2002), 166–173
- [1987] Karp, R.M., Leighton, F.T., Rivest, R.L., Thompson, C.D., Vazirani, U.V., and Vazirani, V.V. [1987]: Global wire routing in two-dimensional arrays. *Algorithmica* 2 (1987), 113–129
- [2001] Leighton, T., Lu, C.-J., Rao, S., and Srinivasan, A. [2001]: New algorithmic aspects of the Local Lemma with applications to routing and partitioning. *SIAM Journal on Computing* 31 (2001), 626–641
- [2002] Müller, D. [2002]: Bestimmung der Verdrahtungskapazitäten im Global Routing von VLSI-Chips. Diploma thesis, University of Bonn, 2002
- [1995] Plotkin, S.A., Shmoys, D.B., and Tardos, É. [1995]: Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research* 20 (1995), 257–301
- [1986] Raghavan, P. [1986]: Randomized rounding and discrete ham-sandwich theorems: provably good algorithms for routing and packing problems. Ph.D. thesis, Report No. UCB/CSD 87/312, University of California, Berkeley 1986
- [1988] Raghavan, P. [1988]: Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences* 37 (1988), 130–143
- [1987] Raghavan, P., and Thompson, C.D. [1987]: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica* 7 (1987), 365–374

- [1991]Raghavan, P., and Thompson, C.D. [1991]: Multiterminal global routing: a deterministic approximation. *Algorithmica* 6 (1991), 73–82
- [2000]Roughgarden, T., and Tardos, É. [2000]: How bad is selfish routing? *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science* (2000), 93–102
- [2003]Schulz, A., and Stier Moses, N. [2003]: Performance of user equilibria in traffic networks. *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms* (2003), 86–87
- [1990]Shahrokhi, F., and Matula, D.W. [1990]: The maximum concurrent flow problem. *Journal of the ACM* 37 (1990), 318–334
- [1994]Vygen, J. [1994]: Disjoint paths. Report No. 94816–OR, Research Institute for Discrete Mathematics, University of Bonn, 1994
- [2003]Xu, J., Hong, X., Jing, T., Cai, Y., and Gu, J. [2003]: A novel timing-driven global routing algorithm considering coupling effects for high performance circuit design. *Proceedings of the Asia and South Pacific Design Automation Conference*, 2003, 847–850
- [1995]Young, N.E. [1995]: Randomized rounding without solving the linear program. *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms* (1995), 170–178

A Flow-Based Method for Improving the Expansion or Conductance of Graph Cuts

Kevin Lang¹ and Satish Rao^{2**}

¹ Yahoo, 210 S. Delacey Ave #105, Pasadena, CA 91105;
langk@overture.com

² CS Dept, Soda Hall; UCB; Berkeley, CA 94720;
satishr@cs.berkeley.edu

Abstract. We discuss the Max-flow Quotient-cut Improvement (MQI) algorithm, a flow-based method for improving graph cuts when cut quality is measured by quotient-style metrics such as expansion or conductance. Given a cut (S, \bar{S}) , this algorithm finds the best improvement among all cuts (S', \bar{S}') such that S' is a strict subset of S . This idea has already been used by theorists to give improved bounds for graph bisection and for finding hierarchical oblivious routing schemes. Here we investigate the *practical* utility of the idea and find that MQI can be combined with Metis to obtain a heuristic graph partitioner that does an extremely good job on classic benchmark graphs, VLSI circuits, and four different tasks from the Information Retrieval domain. We also find empirically that Metis+MQI runs in nearly linear time, so it is applicable to very large problems.

1 Practical Motivation

The starting point for this research was a real clustering task arising in Overture’s pay-per-click search business; the marketing department wanted to use clusters to identify additional phrases that advertisers could be encouraged to bid on. This paper is specifically about the task of finding graph cuts with low expansion or conductance. Kannan *et al* [23] provide a good explanation of the relevance of this task to clustering. Here we just state that top-down recursive bi-partitioning is a standard method for clustering. If the data is represented by a graph, and we choose to use a quotient-style objective function, then we have arrived at the present topic.

2 Related Work

Graph partitioning has been extensively studied. Bisection and numerous variants including finding quotient cuts can be shown to be NP-hard using a reduction by Garey, Johnson and Stockmeyer [17]. In 1988, Leighton and Rao developed an $O(\log n)$ approximation algorithm using multicommodity flow [29].

** Research partially supported by NSF award CCR-0105533.

In 2004, Arora, Rao, and Vazirani improved this to $O(\sqrt{\log n})$ by analyzing an SDP relaxation of the problem [6]. Numerous applications of approximation algorithms have been developed over the years [7,30]. Krauthgamer and Feige [14] gave an approximation algorithm for the originally posed question of graph bisection using an approximation algorithm for quotient cut as an important subroutine. Approaches based on convex optimization and spectral methods were shown to be effective on random graphs by Boppana [8], and dense instances of graph partitioning can be well solved according to work of Arora and Karger using sampling techniques. Other theoretical work includes using spectral methods to address the converse problem of ensuring that no small cuts exist, i.e., that is, for certifying expansion [3,33].

Graph partitioning and graph clustering have found practical application in numerous disciplines ranging from VLSI [5], to parallel computing [21], to information retrieval [12]. The specific problem of finding quotient cuts has also found application, for example, in circuit partitioning [19]. Shi and Malik proposed the use of graph partitioning with a measure called normalized cut³ for image segmentation [31].

Practical approaches to partitioning have included local improvement methods such as simulated annealing (evaluated in [22]), Kernighan-Lin [26] and Fiduccia-Mattheyses [15]. Spectral partitioning was first suggested by Donath and Hoffman [13] and is now implemented in numerous programs. The currently dominant practical method for graph partitioning applies local search to successive levels of a multi-resolution representation of the graph. This approach is used by several graph partitioning packages, including Metis in its various incarnations [25,1].

The present work is closely related to the parameterized cut problem studied by Gallo, Grigoriadis and Tarjan [16] and to an application on finding a densest subgraph by Goldberg [18].⁴ Our max-flow construction has been used previously in a theoretical context by Krauthgamer and Feige [14] to give approximations to bisection, and more recently to give improved hierarchical constructions of oblivious routing schemes [20].

3 Max Flow Quotient Cut Improvement (MQI)

In this section we describe a fast exact algorithm for improving either the expansion or the conductance of cuts of weighted or unweighted graphs. For simplicity, we will specifically talk about expansion (*aka* the quotient cut metric) and unweighted graphs.

Our input is an undirected graph and an initial cut, that is, an initial partitioning of the graph's nodes into two sets A and $B = \overline{A}$. WLOG, $(a = |A|) \leq |B|$. There are c edges crossing the cut, so the initial quotient cut score is c/a . Figure 1-left is a schematic diagram of the flow problem that we will build. Note that

³ This notion was previously referred to as conductance in the theory of rapidly mixing Markov chains, see for example, [32].

⁴ In fact, we take a slightly different approach which appears to be sufficient.

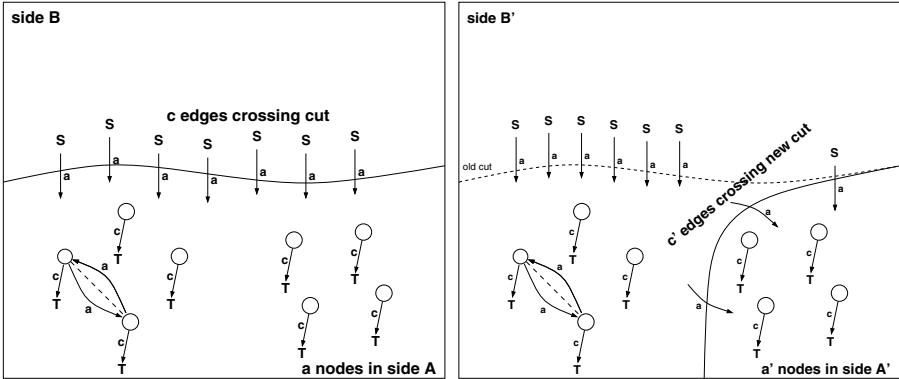


Fig. 1. Setting up the max flow problem, and interpreting its solution.

although this problem will be based on our undirected input graph and cut, we are building a completely new directed graph here. It is convenient to specify this new graph as the output of the following procedure that begins with a copy of the input graph.

1. Discard all B -side nodes.
2. Discard every edge that used to connect a pair of B -side nodes.
3. Replace every edge that used to connect a pair of A -side nodes with a pair of directed edges, one pointing in each direction, and each with capacity a .
4. Add a source node S and a sink node T .
5. Discard each edge that used to connect a B -side node with an A -side node x , replacing it with a single directed edge from S to x , with capacity a .
6. Add a single directed edge from every A -side node to T , with capacity c .

3.1 Analysis

We have an input graph, an initial quotient cut (A, B) (with $|A| \leq |B|$), the max flow problem built above, and a solution to this max flow problem consisting of the total value of the flow plus the amount of flow traversing each edge. WLOG we assume that the solution is in a canonical form with no flow in any edge whose opposite-direction counterpart is saturated. Figure 1-right is intended as a guide to the following arguments.

Theorem 1. *There is an improved quotient cut (A', B') (with A' a strict subset of A) if and only if the maximum flow is less than ca .*

Proof. Forward direction: We start by assuming the existence of an improved quotient cut (A', B') , with A' a strict subset of A . Let us say that A' contains a' nodes and that c' edges cross the improved cut. Then by the definition of quotient cuts, $\frac{c'}{a'} < \frac{c}{a}$.

The net flow into A' on non-sink edges is at most $c'a$, but a flow of $a'c$ would be required to saturate the a' sink edges of A' . The above inequality implies that $c'a < a'c$, so insufficient flow is available, and some sink edge leading out of A' must be unsaturated. But if any sink edge is unsaturated, the total flow into the sink must be less than ca , and our claim is proved. \square

Proof. **Reverse direction:** Here we assume that the maximum flow is less than ca . We will construct a new cut and prove that it has an improved quotient cut score.

The small side A' of the new cut is the set of nodes reachable from the sink by a depth-first search along reversed unsaturated edges. This “backwards DFS” procedure advances from a node y to a node x if and only if there is an unsaturated edge leading from node x to node y .

The analysis of the new cut begins by noting that the total capacity of all sink edges is ac , while the total flow is less than ca , so at least one sink edge must be unsaturated. Therefore at least one node of A is reachable by the backwards DFS, so A' is nonempty, and we will be able to divide by $a' = |A'|$ a little later in the proof. Also, the existence of at least one unsaturated edge leading from A' to the sink means that F_s , the total flow from A' to the sink, must be less than $a'c$.

Now, since we built A' by going as far back as we could along unsaturated edges, every edge feeding into A' from a non- A' node must be saturated. Let us say there are c' such edges. Then F_i , the flow into A' , is exactly $c'a$.

Recall that the flow is in a canonical form with no flow contrary to any saturated edge. Therefore F_o , the flow out of A' , is entirely carried by sink edges, so $F_o = F_s$. Finally, by conservation of flow, $F_i = F_o$. Combining these facts, $c'a = F_i = F_o = F_s < a'c$. Because $a \geq 1$ and $a' \geq 1$, we can divide to get $\frac{c'}{a'} < \frac{c}{a}$, which proves that (A', B') is a better quotient cut than (A, B) . \square

3.2 Finding the Best Reachable Quotient Cut

A single call to MQI determines whether any reachable improved quotient cut exists, and if so, it returns one of them. We can find the *best* reachable quotient cut in about $\log c + \log a$ MQI runs via binary search.⁵ Together with the NP-hardness of the quotient cut problem, this fact yields the following:

Corollary 1. *The problem of finding any cut whose small side contains the small side of the best quotient cut is NP-hard.*

Despite this worst-case intractability result, the modified problem mentioned in this corollary intuitively seems easier than the original quotient cut problem, and the success of the partitioning system discussed below is empirical evidence that it is somewhat tractable in practice.

⁵ Instead of doing binary search, our code repeatedly feeds MQI’s output back into its input until it announces that no further improvement is possible. This feedback approach empirically seems to require fewer calls to MQI.

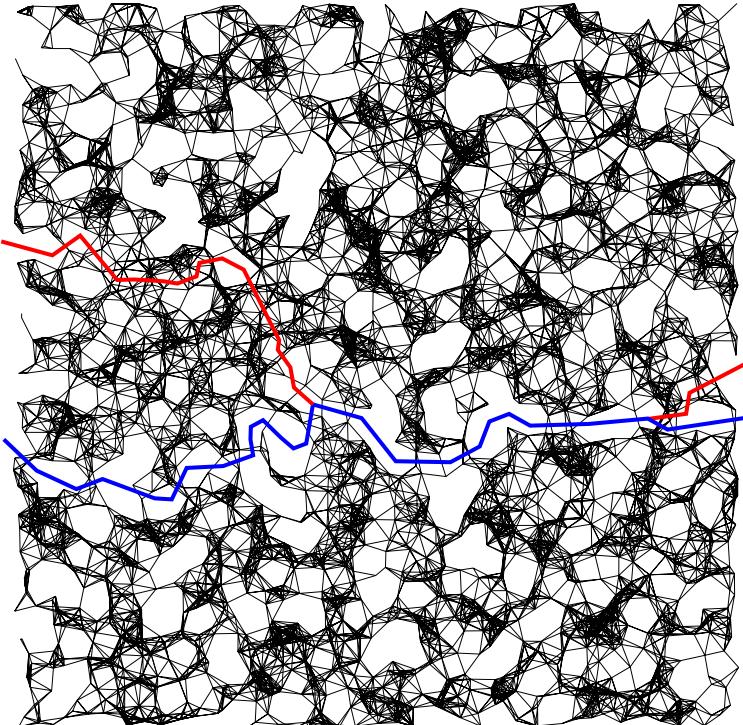


Fig. 2. Cuts found by Metis (red) and MQI (blue,lower) in graph U3A.

3.3 Multi-try Metis+MQI

Because MQI always reduces the balance of a partition as a side effect of improving its quotient cut score, a maximally balanced partition, that is, a bisection, is a good starting point. Therefore we use Metis [25], a fast implementation of multi-resolution Fiduccia-Mattheyses [15], to obtain initial balanced cuts that are fed into MQI.

Since MQI cannot reach all cuts from any particular starting cut (because it can only move nodes out of the small side), it seems likely that our chances of finding an optimal solution would be improved by multiple runs from different starting cuts. Therefore we perform multiple tries of Metis+MQI with different RNG seeds for Metis.

We expect the empirical growth rate of this system's run time to be not much worse than linear, because Metis and the max-flow solver `hi_pr` (see [11]) both run in nearly linear time in practice; our MQI feedback loop seems to require a sublogarithmic number of iterations; and our decision to do multiple tries only affects the constant factor.

graph	BFS+FMq			Spectral Method			Leighton-Rao			Metis+MQI		
	c/a	c	a	c/a	c	a	c/a	c	a	c/a	c	a
U3A	.0937	138	1473	.0613	78	1272	.0515	61	1184	.0515	61	1184
U3B	.0617	92	1491	.0437	63	1442	.0358	13	363	.0323	48	1484
U3C	.1058	152	1437	.0940	128	1362	.0780	101	1295	.0754	112	1486
U3D	.1677	237	1413	.2461	159	646	.1626	207	1273	.1469	220	1498
U3E	.0767	112	1461	.0789	84	1065	.0500	73	1460	.0433	63	1455
U10A	.0665	330	4965	.0579	249	4298	.0424	204	4811	.0411	198	4821
U10B	.0431	214	4970	.0391	136	3474	.0301	117	3883	.0296	135	4554
U10C	.0552	274	4962	.0458	203	4435	.0343	155	4522	.0343	155	4524
U10D	.0874	437	4999	.0714	315	4409	.0575	284	4937	.0568	275	4840
U10E	.0573	285	4976	.0467	212	4543	.0291	143	4908	.0292	142	4860
U30A	.0402	587	14596	.0377	458	12135	.0249	335	13467	.0246	340	13844
U100K	.0255	1265	49573	.0218	972	44581	.0156	692	44370	.0150	723	48176

Fig. 3. Metis+MQI and Leighton-Rao crush BFS+FMq and Spectral on geometric graphs [the best cuts are shown in bold].

3.4 Generalization of MQI to Weighted Graphs, and to Conductance

The MQI method outlined above extends in a straightforward way to graphs with weights on the edges and nodes. Also, if the weight on each node is set to be the sum of the weights of the node’s incident edges, then MQI is trying to optimize conductance, a metric that has been found by [23], by [31], and also by us to be more robust than expansion for clustering tasks.

4 Experiment: Geometric Graphs

Geometric graphs, introduced by Johnson *et al* in [22], are a widely used benchmark because they are easy to visualize but hard to partition. A geometric graph is generated by randomly choosing points in the unit square, and then connecting all pairs of points that are closer than a distance threshold (see figure 2). The experiments in [22] showed that Kernighan-Lin [26] works better than other local search algorithms (*e.g.* simulated annealing) on small geometric graphs.

In [28], Lang and Rao found that on somewhat larger geometric graphs, Leighton-Rao (which looks for congested edges in the solution to a multicommodity flow problem) finds much better solutions than local search (specifically FMq, a quotient cut version of Fiduccia-Mattheyses [15]).

In the current experiment we use the quotient cut (*aka* expansion) metric for unweighted graphs, namely $c/(\min(|A|, |B|))$ for a cut containing c edges that partitions the nodes into the sets A and B . Results for four different algorithms on graphs with 3,000 to 100,000 nodes are shown in Table 3.

As in [28], we see that Leighton-Rao decisively beats BFS+FMq. The spectral method also performs significantly worse than Leighton-Rao, but the new

Metis+MQI system actually works a little bit better. Implementation details for the four methods can be found in [27].

5 Experiments: Finding Low Conductance Cuts while Clustering Data from Four IR Domains

As mentioned in the introduction, the immediate motivation for this research was a real large-scale clustering task at Overture. Here we test the Metis+MQI system on the Overture task plus three other Information Retrieval datasets to show that the method has general applicability.

The rationale for our experimental methodology can be presented succinctly by referring to two papers by Kannan, Vempala, et al. In [23], theoretical arguments are presented for the following two contentions. First, a good approach to clustering a data graph is to recursively bi-partition it using low-conductance cuts. Second, the spectral method is a good heuristic method for finding low-conductance cuts. In [10], a complete clustering system is built along these lines, evaluated on about eight benchmark tasks from the clustering literature, and found to be very good in end-to-end clustering tests including pre- and post-processing that carefully matches the corresponding processing in the respective earlier papers.

To avoid the laborious and tricky task of accurately reproducing all of this machinery, we make the following argument: in a clustering system built along the lines of [23] and [10], the subroutine for finding low-conductance cuts is basically a black-box. While the black box in those papers uses the spectral method, one could easily substitute a black box that uses Metis+MQI, and if this new black box does at least as good a job of finding low conductance cuts of subgraphs that arise during a clustering task, then the overall system should still work fine.

In these experiments we will be using the conductance metric for unweighted graphs, namely $c/(\min(D_A, D_B))$, to evaluate a cut containing c edges that partitions the nodes into sets A and B , and where $D_X = \sum_{i \in X} d_i$ is the sum of the degrees of nodes in the set X .

The input graphs in all of these four tasks are bipartite, and can be thought of as representing preference or occurrence matrices. In one there are advertisers bidding on phrases. In another there are actresses appearing in movies. In another there are words appearing in news stories. In another there are web pages linking to other web pages. Although the graphs are bipartite, we partition them as if they aren't, resulting in clusters containing a mixture of elements from both sides. The result can be interpreted as a co-clustering of the two sides.

Our performance comparisons are with a Spectral code written by Leonid Zhukov of Overture. It incorporates time and space saving optimizations for bipartite input graphs (see [12]). The eigenproblems are solved by ARPACK (an industrial-strength implementation of the Lanczos method) using 10 Arnoldi vectors. The Spectral code tries to optimize the normalized cut metric of [31], which is very similar to conductance, but has a product rather than a minimum

in the denominator. The values in the resulting second eigenvector imply an ordering of the graph's N nodes; this ordering implies a choice of $N-1$ cuts, and we choose the one with the best conductance.

5.1 Overture Advertisers and Phrases

Overture serves as the web search back-end for many Internet portals. It makes money by soliciting bids from advertisers on various search phrases. When a search engine user types in one of these phrases, an auction occurs, and the web pages of the highest bidders appear near the top of the search results. If the user clicks on one of these listings, the respective advertiser pays Overture the amount of their bid. The complete bidding database contains hundreds of thousands of advertisers and millions of phrases and bids. The experiments of this section employ a medium-sized subset of this database containing about 150,000 phrases, 120,000 advertisers and 3 million bids. [More details can be found in [9].]

Figure 5A shows a scatter plot of the conductance of cuts found by Metis+MQI and Spectral on various subsets of this data.⁶ Clearly Metis+MQI did very well; its win-loss-tie record with 3 tries is 93-2-20. Figure 5B shows how run time varies with subgraph size on this task. The growth rate is roughly similar for the two methods, but the constant factor of Metis+MQI is worse, partly due to our use of multiple tries. We note that a single try already yields a win-loss-tie of record of 88-9-18. In actual use we are willing to pay for lots of tries to get the cleanest possible results. On the 115 subgraphs in this benchmark set, 11 tries gets Metis+MQI down to zero losses to Spectral.

5.2 Notre-Dame Web Pages

The Notre-Dame web graph is one of the examples in [2], a classic paper on the statistics of the web. We downloaded this directed graph from the authors' website, and used it to build an undirected bipartite graph reflecting its link structure. For each node in the web graph, there are two nodes in the bipartite graph, one on the left-hand "from" side, and one on the right-hand "to" side. For each arc in the web graph, there is one edge in the bipartite graph showing which "from" node is connected to which "to" node. After eliminating degree 1 nodes, we ended up with a bipartite graph containing 234,099 nodes and 1,272,723 edges. Figure 5C shows that Metis+MQI and Spectral found similar cuts. The former's win-loss-tie record with 6 tries is 7-0-32.

The run times seen in Figure 5D are more interesting: although the constant factor is again somewhat higher for Metis+MQI (6 tries) than for Spectral, the asymptotic growth of Spectral's run time is much worse, causing the curves to

⁶ For all four tasks, the subgraphs for testing cut quality were obtained by recursively bi-partitioning the original graph using BFS+FMq, down to a subgraph size of 5000 nodes. The subgraphs for testing run time scaling were obtained by repeatedly shaving off 10 percent of the nodes using Metis.

cross at a graph size of about 100,000. As usual, the growth rate for Metis+MQI is a bit over linear, perhaps $n \log n$ or $n^{1.2}$, while the run time for Spectral grows quadratically.⁷

5.3 IMDB Actresses and Movies

Starting from raw data downloaded from IMDB, we built a bipartite graph showing which actresses appeared in which movies over the past 90 years. After eliminating degree-1 nodes, this graph contained 254,462 nodes and 782,142 edges.

Figure 5E shows that once again Metis+MQI (3 tries) provides good cuts, with a win-loss-tie record of 50-0-36 against Spectral. In figure 5F, we see that the run time of Metis+MQI has a higher constant factor but a lower growth rate than Spectral. The run time of Spectral blows up worse than on the Overture bidding data, but not as badly as on the Notre-Dame web data.

5.4 AP News Stories and Words

The raw data for this task is the TREC1 1989 AP news stories dataset. There are over 84,000 articles containing about 170,000 different words. We counted occurrences of words in each article, computed totals for the whole year, and discarded any words that occurred less than 10 or more than 70,000 times for the year. Then, for each article, we chose the 10 words that had the largest ratio of occurrences in the document to occurrences in the year. Finally, we build a bipartite occurrence graph with the articles on one side and the chosen words on the other. After eliminating degree-1 nodes, there were 133,683 nodes and 845,462 edges.

Figures 5G and 5H show the cut scores and run times of Metis+MQI and Spectral on this task. The win-loss-tie record of Metis+MQI (3 tries) is 16-0-30. The run time constant factor for Metis+MQI is a bit higher than for Spectral. The run time growth rate is low for both algorithms, with Spectral apparently a bit worse.

5.5 Summary of Results

In these four IR domains, Metis+MQI compares very favorably to the spectral method in its ability to find low conductance cuts. Overall, Metis+MQI had a total of 166 wins, 2 losses, and 118 ties on these test graphs. The run time of Metis+MQI had a somewhat higher constant factor, partly due to our use of multiple tries, but on all four tasks, its run time growth rate was not much worse than linear. The growth rate for Spectral was less consistent, ranging from nearly linear to nearly quadratic.

⁷ We note that the cuts in this graph are very deep. This fact probably contributes both to Spectral's good performance and also to its slow convergence.

graph	nodes	hMetis +MQI (bal=10)	hMetis (3 tries) (newbal)	win-loss-tie	graph	nodes	hMetis +MQI (bal=10)	hMetis (3 tries) (newbal)	win-loss-tie
ibm01	12 K	.0246	.0253	3-0	ibm10	69 K	.0346	.0346	0-0-3
ibm02	19 K	.0211	.0209	0-3	ibm11	71 K	.0276	.0274	2-1
ibm03	23 K	.0601	.0758	3-0	ibm12	71 K	.0420	.0420	1-0-2
ibm04	27 K	-	-	-	ibm13	84 K	.0068	.0068	2-0-1
ibm05	29 K	.1353	.1355	2-0-1	ibm14	147 K	.0141	.0141	2-0-1
ibm06	32 K	.0578	.0565	0-3	ibm15	161 K	.0066	.0066	0-3
ibm07	45 K	.0388	.0387	0-0-3	ibm16	183 K	.0206	.0205	2-1
ibm08	51 K	.0445	.0445	0-0-3	ibm17	185 K	.0261	.0270	3-0
ibm09	54 K	.0234	.0237	3-0	ibm18	211 K	.0148	.0170	3-0

Fig. 4. hMetis+MQI frequently beats 3 runs of hMetis on VLSI circuits (hypergraphs) of the ISPD98 benchmark suite.

6 Experiment: VLSI Circuits

VLSI circuits are hypergraphs rather than graphs, but a quotient cut can be defined as the problem of minimizing the number of hyperedges split by the cut divided by the number of nodes on the small side. A form of the Metis methodology for this task was implemented in the program hMetis [24]. Via standard reductions from hyperedge partitioning to node partitioning and then to directed edge partitioning, MQI can be translated to optimize hyperedge quotient cut cost rather than edge cut cost.

For this problem, we do not as yet have a Spectral partitioning code or an implementation of Leighton-Rao, so we are limited in the comparisons that we do. On the other hand, we can compare with the hypergraph partitioner hMetis, which (unlike Metis) can generate relaxed-balance solutions, with the amount of “slack” controllable through a balance parameter. For example, with balance parameter of 10, hMetis attempts to find the smallest cut where the small side contains at least 40 percent of the nodes.

In our experiments, we ran hMetis with balance parameter 10, and then used the hypergraph formulation of MQI to optimize both sides of this hMetis cut. Assume that (S, \bar{S}) was the cut of smallest quotient that was found during this step.

We then reran hMetis three times, with the balance parameter set to the balance of the cut (S, \bar{S}) .⁸ That is, (S, \bar{S}) is a legal output for these runs of hMetis. We did not run MQI on the output of the three successive runs of hMetis so that we can compare the state of the art against the state of the art plus MQI.

In table 4, we give the results for the ISPD 98 benchmark suite of circuits [4]. For each graph, we report the number of nodes, the quotient cost of

⁸ We note that MQI is much faster than hMetis, so running hMetis twice and choosing the best is almost twice as slow as running hMetis and then MQI.

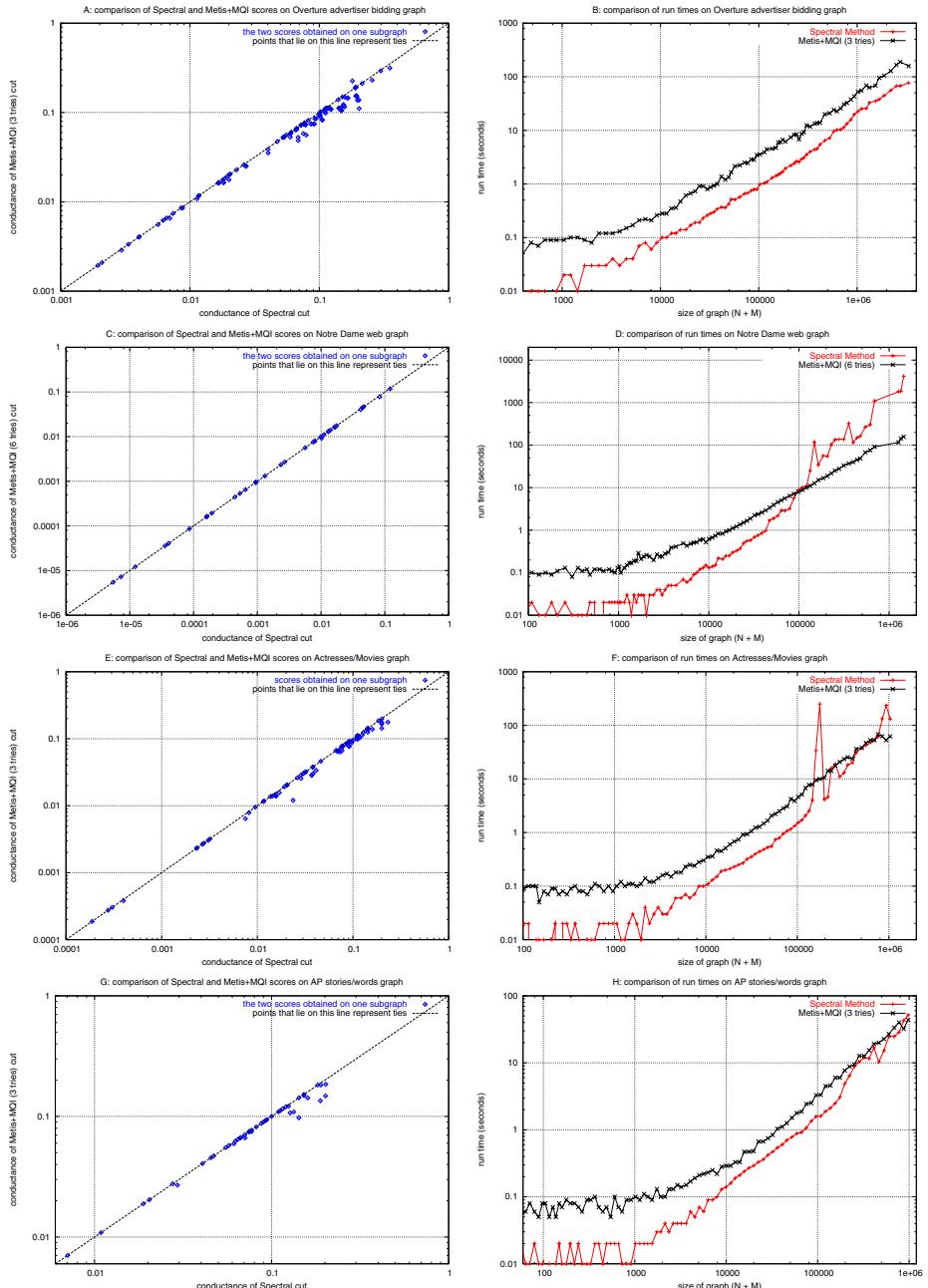


Fig. 5. On four IR tasks, Metis+MQI usually beats or ties Spectral in both cut quality and asymptotic run time growth.

hMetis(bal=10) plus MQI, the best quotient cost of three runs of hMetis(newbal), and the win-loss-tie record of the one run of hMetis(10) plus MQI against the three runs of hMetis(newbal). Again, newbal is set to the balance of the cut output by hMetis(bal=10) plus MQI.⁹

The differences revealed by this VLSI experiment are mostly rather small. Still, MQI often produces better solutions overall, and wins more often than not. On the other hand, it seems that for all but the very largest graphs, the techniques in hMetis can (eventually) find as good a solution as the quicker MQI improved hMetis approach. Thus, we suggest using MQI since it is cheap and in combination gives some improvement over not using it. Also, for the largest two graphs, we should note that MQI found quotient cuts that were quite a bit better than hMetis alone.

References

1. R. Aggarwal, G. Karypis, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. *IEEE Transactions on VLSI Systems*, 7(1), 1999.
2. R. Albert, A. Barabasi, and H. Jeong. Diameter of the world wide web. *Nature*, 401:130–131, 1999.
3. N. Alon and V.D. Milman. 1 isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38:73–88, 1985.
4. C. J. Alpert. The ispd98 circuit benchmark suite. In *Proceedings International Symposium on Physical Design*, pages 85–90, April 1998.
5. C. J. Alpert and A. B. Kahng. Recent developments in netlist partitioning: A survey. *VLSI Journal*, 19(1):1–81, 1995.
6. Sanjeev Arora, Satish Rao, Umesh Vazirani. Expander flows and a $O(\sqrt{\log n})$ approximation to sparsest cut. *STOC*, 2004.
7. S.N. Bhatt and F.T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):303–343, 1984.
8. Ravi B. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Symposium on Foundations of Computer Science(FOCS)*, pages 280–285, 1987.
9. John Joseph M. Carrasco, Dan Fain, Kevin Lang, and Leonid Zhukov. Clustering of bipartite advertiser-keyword graph. Technical report, OR-2003-017, Overture/Yahoo Labs, 2003.
10. David Cheng, Ravi Kannan, Santosh Vempala, and Grant Wang. On a recursive spectral algorithm for clustering from pairwise similarities. Technical report, MIT-LCS-TR-906, MIT, 2003.
11. Boris V. Cherkassky and Andrew V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.
12. Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.
13. W.E. Donath and A. J. Hoffman. Lower bounds for partitioning of graphs. *IBM J. Res. Develop.*, 17:420–425, 1973.

⁹ The ibm04 graph got no benefit from MQI, so we reported nothing further.

14. Uriel Feige and Robert Krauthgamer. A polylogarithmic approximation of the minimum bisection. In *Symposium on the Foundations of computer science*, pages 105–115, 2000.
15. C.M. Fiduccia and R.M. Mattheyses. A linear time heuristic for improving network partitions. In *Design Automation Conference*, pages 175–181, 1982.
16. G. Gallo, M.D. Grigoriadis, and R.E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18:30–55, 1989.
17. M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, pages 237–267, 1976.
18. Andrew V. Goldberg. Finding a maximum density subgraph. Technical report, UCB CSD 84/71, University of California, Berkeley, 1984.
19. L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. on CAD*, 11(9):1074–1085, September 1992.
20. Chris Harrelson, Kirsten Hildrum, and Satish Rao. A polynomial-time tree decomposition to minimize congestion. In *Proceedings of the Fifteenth ACM Symposium on Parallelism Algorithms and Architectures*, June 2003.
21. Bruce Hendrickson and Robert Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. Sci. Stat. Comput.*, 16(2):452–469, 1995.
22. D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Shevon. Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations Research*, 37(6):865–892, 1989.
23. Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings - good, bad and spectral. In *Symposium on Foundations of Computer Science(FOCS)*, 1999.
24. George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Applications in VLSI domain. Technical report, UMN, 1997.
25. George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359 – 392, 1999.
26. B. W. Kernighan and S. Lin. An ecient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 29(2):291–307, 1970.
27. Satish Rao Kevin Lang. Improving quotient cuts with max flow. Technical report, OR-2003-014, Overture/Yahoo Labs, 2003.
28. Kevin Lang and Satish Rao. Finding near-optimal cuts: An empirical evaluation. In *Symposium on Discrete Algorithms*, pages 212–221, 1993.
29. F. T. Leighton and Satish Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *28th Annual Symposium on Foundations of Computer Science, IEEE*, pages 256–269, 1988.
30. Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6):787–832, 1999.
31. Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
32. Alistair Sinclair. Improved bounds for mixing rates of markov chains and multi-commodity flow. *Combinatorics, Probability and Computing*, 1:351–370, 1992.
33. R.M. Tanner. Explicit concentrators from generalized n-gons. *SIAM J. Alg. Disc. Methods*, 5:287–293, 1984.

All Rational Polytopes Are Transportation Polytopes and All Polytopal Integer Sets Are Contingency Tables

Jesus De Loera¹ * and Shmuel Onn² **

¹ University of California at Davis, Davis, CA 95616, USA

deloera@math.ucdavis.edu, <http://www.math.ucdavis.edu/~deloera>

² Technion - Israel Institute of Technology, 32000 Haifa, Israel

onn@ie.technion.ac.il <http://ie.technion.ac.il/~onn>

Abstract. We show that any rational polytope is polynomial-time representable as a “slim” $r \times c \times 3$ three-way line-sum transportation polytope. This universality theorem has important consequences for linear and integer programming and for confidential statistical data disclosure. It provides polynomial-time embedding of arbitrary linear programs and integer programs in such slim transportation programs and in bipartite biflow programs. It resolves several standing problems on 3-way transportation polytopes. It demonstrates that the range of values an entry can attain in any slim 3-way contingency table with specified 2-margins can contain arbitrary gaps, suggesting that disclosure of k -margins of d -tables for $2 \leq k < d$ is confidential.

Our construction also provides a powerful tool in studying concrete questions about transportation polytopes and contingency tables; remarkably, it enables to automatically recover the famous “real-feasible integer-infeasible” $6 \times 4 \times 3$ transportation polytope of M. Vlach, and to produce the first example of 2-margins for $6 \times 4 \times 3$ contingency tables where the range of values a specified entry can attain has a gap.

1 Introduction

Transportation polytopes, their integer points (contingency tables), and their projections, have been used and studied extensively in the operations research and mathematical programming literature, see e.g. [1,2,3,12,14,17,18,21,22] and references therein, and in the context of secure statistical data management by agencies such as the U.S. Census Bureau [20], see e.g. [5,6,9,10,13,16] and references therein.

We start right away with the statement of the main theorem of this article, followed by a discussion of some of its consequences for transportation polytopes,

* Supported in part by NSF grant 0309694 and by AIM - the American Institute of Mathematics.

** Supported in part by a grant from ISF - the Israel Science Foundation, by AIM - the American Institute of Mathematics, by the Technion President Fund, and by the Fund for the Promotion of Research at the Technion.

linear and integer programming, bipartite biflows, and confidential statistical data disclosure. Throughout, \mathbb{R}_+ denotes the set of nonnegative reals.

Theorem 1. *Any rational polytope $P = \{y \in \mathbb{R}_+^n : Ay = b\}$ is polynomial-time representable as a “slim” (of smallest possible depth three) 3-way transportation polytope*

$$T = \{x \in \mathbb{R}_+^{r \times c \times 3} : \sum_i x_{i,j,k} = w_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\}.$$

By saying that a polytope $P \subset \mathbb{R}^p$ is *representable* as a polytope $Q \subset \mathbb{R}^q$ we mean in the strong sense that there is an injection $\sigma : \{1, \dots, p\} \rightarrow \{1, \dots, q\}$ such that the coordinate-erasing projection

$$\pi : \mathbb{R}^q \rightarrow \mathbb{R}^p : x = (x_1, \dots, x_q) \mapsto \pi(x) = (x_{\sigma(1)}, \dots, x_{\sigma(p)})$$

provides a bijection between Q and P and between the sets of integer points $Q \cap \mathbb{Z}^q$ and $P \cap \mathbb{Z}^p$. In particular, if P is representable as Q then P and Q are isomorphic in any reasonable sense: they are linearly equivalent and hence all linear programming related problems over the two are polynomial-time equivalent; they are combinatorially equivalent hence have the same facial structure; and they are integer equivalent and hence all integer programming and integer counting related problems over the two are polynomial-time equivalent as well.

The polytope T in the theorem is a 3-way transportation polytope with specified *line-sums* $(u_{i,j}), (v_{i,j}), (w_{i,j})$ (2-margins in the statistical context to be elaborated upon below). The arrays in T are of size $(r, c, 3)$, that is, they have r rows, c columns, and “slim” depth 3, which is best possible: 3-way line-sum transportation polytopes of depth ≤ 2 are equivalent to ordinary 2-way transportation polytopes which are not universal.

An appealing feature of the representation manifested by Theorem 1 is that the defining system of T has only $\{0, 1\}$ -valued coefficients and depends only on r and c . Thus, every rational polytope has a representation by one such system, where all information enters through the right-hand-side data $(u_{i,j}), (v_{i,j}), (w_{i,j})$.

We now discuss some of the consequences of Theorem 1.

1.1 Universality of Transportation Polytopes: Solution of the Vlach Problems

As mentioned above, there is a large body of literature on the structure of various transportation polytopes. In particular, in the comprehensive paper [21], M. Vlach surveys some ten families of necessary conditions published over the years by several authors (including Schell, Haley, Smith, Moràvek and Vlach) on the line-sums $(u_{i,j}), (v_{i,j}), (w_{i,j})$ for a transportation polytope to be nonempty, and raises several concrete problems regarding these polytopes. Specifically, [21, Problems 4,7,9,10] ask about the sufficiency of some of these conditions. Our results show that transportation polytopes (in fact already of slim $(r, c, 3)$ arrays) are universal and include all polytopes. This indicates that the answer to each

of Problems 4,7,9,10 is negative. Details will appear elsewhere. Problem 12 asks whether all dimensions can occur as that of a suitable transportation polytope: the affirmative answer, given very recently in [10], follows also at once from our universality result.

Our construction also provides a powerful tool in studying concrete questions about transportation polytopes and their integer points, by allowing to write down simple systems of equations that encode desired situations and lifting them up. Here is an example to this effect.

Example 1. Vlach's rational-nonempty integer-empty transportation: using our construction, we automatically recover the smallest known example, first discovered by Vlach [21], of a rational-nonempty integer-empty transportation polytope, as follows. We start with the polytope $P = \{y \geq 0 : 2y = 1\}$ in one variable, containing a (single) rational point but no integer point. Our construction represents it as a transportation polytope T of $(6, 4, 3)$ -arrays with line-sums given by the three matrices below; by Theorem 1, T is integer equivalent to P and hence also contains a (single) rational point but no integer point.

$$(u_{i,j}) = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad (v_{i,k}) = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad (w_{j,k}) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Returning to the Vlach problems, [21, Problem 13] asks for a characterization of line-sums that guarantees an integer point in T . In [13], Irving and Jerrum show that deciding $T \cap \mathbb{Z}^{r \times c \times h} \neq \emptyset$ is NP-complete, and hence such a characterization does not exist unless P=NP. Our universality result strengthens this to *slim* arrays of smallest possible constant depth 3; we have the following immediate corollary of Theorem 1.

Corollary 1. *Deciding if a slim $r \times c \times 3$ transportation polytope has an integer point is NP-complete.*

A comprehensive complexity classification of this decision problem under various assumptions on the array size and on the input, as well as of the related counting problem and other variants, are in [4].

The last Vlach problem [21, Problem 14] asks whether there is a *strongly-polynomial-time* algorithm for deciding the (real) feasibility $T \neq \emptyset$ of a transportation polytope. Since the system defining T is $\{0, 1\}$ -valued, the results of Tardos [19] provide an affirmative answer. However, the existence of a strongly-polynomial-time algorithm for linear programming in general is open and of central importance; our construction embeds any linear program in an $r \times c \times 3$ transportation program in polynomial-time, but unfortunately this process is *not* strongly-polynomial. Nonetheless, our construction may shed some light on the problem and may turn out useful in sharpening the boundary (if any) between strongly and weakly polynomial-time solvable linear programs.

1.2 Universality of Bipartite Biflows

Another remarkable immediate consequence of Theorem 1 is the universality of the two-commodity flow (biflow) problem for bipartite digraphs where all arcs are oriented the same way across the bipartition. The problem is the following: given *supply* vectors $s^1, s^2 \in \mathbb{R}_+^r$, *demand* vectors $d^1, d^2 \in \mathbb{R}_+^c$, and *capacity* matrix $u \in \mathbb{R}_+^{r \times c}$, find a pair of nonnegative “flows” $x^1, x^2 \in \mathbb{R}_+^{r \times c}$ satisfying supply and demand requirements $\sum_j x_{i,j}^k = s_i^k$, $\sum_i x_{i,j}^k = d_j^k$, $k = 1, 2$, and capacity constraints $x_{i,j}^1 + x_{i,j}^2 \leq u_{i,j}$. In network terminology, start with the directed bipartite graph with vertex set $R \sqcup C$, $|R| = r$, $|C| = c$, and arc set $R \times C$ with capacities $u_{i,j}$, and augment it with two sources a^1, a^2 and two sinks b^1, b^2 , and with arcs (a^k, i) , $i \in R$, (j, b^k) , $j \in C$, $k = 1, 2$ with capacities $u(a^k, i) := s_i^k$, $u(j, b^k) := d_j^k$. We have the following universality statement.

Corollary 2. *Any rational polytope $P = \{y \in \mathbb{R}_+^n : Ay = b\}$ is polynomial-time representable as a bipartite biflow polytope*

$$\begin{aligned} F = \{ (x^1, x^2) \in \mathbb{R}_+^{r \times c} \oplus \mathbb{R}_+^{r \times c} : & x_{i,j}^1 + x_{i,j}^2 \leq u_{i,j}, \\ & \sum_j x_{i,j}^k = s_i^k, \quad \sum_i x_{i,j}^k = d_j^k, \quad k = 1, 2 \}. \end{aligned}$$

Proof. By an easy adjustment of the construction in §2.3 used in the proof of Theorem 3 (see §2.3 for details): take the capacities to be $u_{I,J}$ as defined in §2.3; take the supplies to be $s_I^1 := V_{I,1} = U$ and $s_I^2 := V_{I,3} = U$ for all I ; and take the demands to be $d_J^1 := W_{J,1}$ and $d_J^2 := W_{J,3}$ for all J . Note that by taking s_I^2 and d_J^2 to be $V_{I,3}$ and $W_{J,3}$ instead of $V_{I,2}$ and $W_{J,2}$ we get that all supplies have the same value U , giving a stronger statement. \square

As noted in the proof of Corollary 2, it remains valid if we take all supplies to have the same value $s_i^k = U$, $i = 1, \dots, r$, $k = 1, 2$; further (see discussion in the beginning of §2.3 on encoding forbidden entries by bounded entries), all capacities $u_{i,j}$ can be taken to be $\{0, U\}$ -valued, giving a stronger statement.

Moreover, as the next example shows, using our construction we can systematically produce simple bipartite biflow acyclic networks of the above form, with integer supplies, demands and capacities, where any feasible biflow must have an arbitrarily large prescribed denominator, in contrast with Hu’s celebrated half-integrality theorem for the undirected case [11].

Example 2. Bipartite biflows with arbitrarily large denominator: Fix any positive integer q . Start with the polytope $P = \{y \geq 0 : qy = 1\}$ in one variable containing the single point $y = \frac{1}{q}$. Our construction represents it as a bipartite biflow polytope F with integer supplies, demands and capacities, where y is embedded as the flow $x_{1,1}^1$ of the first commodity from vertex $1 \in R$ to $1 \in C$. By Corollary 2, F contains a single biflow with $x_{1,1}^1 = y = \frac{1}{q}$. For $q = 3$, the data for the biflow problem is below, resulting in a unique, $\{0, \frac{1}{3}, \frac{2}{3}\}$ -valued, biflow.

$$(u_{i,j}) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}, \quad (s_i^1) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad (s_i^2) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix},$$

$$(d_j^1) = (1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0), \quad (d_j^2) = (0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 2 \ 1).$$

By Corollary 2, any (say, feasibility) linear programming problem can be encoded as such a bipartite biflow problem (unbounded programs can also be treated by adding to the original system a single equality $\sum_{j=0}^n y_j = U$ with y_0 a new “slack” variable and U derived from the Cramer’s rule bound). Thus, any (hopefully combinatorial) algorithm for such bipartite biflow will give an algorithm for general linear programming. There has been some interest lately [15] in combinatorial approximation algorithms for (fractional) multiflows, e.g. [7,8]; these yield, via Corollary 2, approximation algorithms for general linear programming, which might prove a useful and fast solution strategy in practice. Details of this will appear elsewhere.

1.3 Confidential Statistical Data Disclosure: Models, Entry-Range, and Spectrum

A central goal of statistical data management by agencies such as the U.S. Census Bureau is to allow public access to as much as possible information in their data base. However, a major concern is to protect the confidentiality of individuals whose data lie in the base. A common practice [5], taken in particular by the U.S. Census Bureau [20], is to allow the release of some margins of tables in the base but not the individual entries themselves. The security of an entry is closely related to the range of values it can attain in any table with the fixed released collection of margins: if the range is “simple” then the entry may be exposed, whereas if it is “complex” the entry may be assumed secure.

In this subsection only, we use the following notation, which is common in statistical applications. A *d-table of size n = (n₁, ..., n_d)* is an array of non-negative integers $x = (x_{i_1, \dots, i_d})$, $1 \leq i_j \leq n_j$. For any $0 \leq k \leq d$ and any k -subset $J \subseteq \{1, \dots, d\}$, the *k-margin* of x corresponding to J is the k -table $x^J := (x_{i_j:j \in J}^J) := (\sum_{i_j:j \notin J} x_{i_1, \dots, i_d})$ obtained by summing the entries over all indices *not in J*. For instance, the 2-margins of a 3-table $x = (x_{i_1, i_2, i_3})$ are its *line-sums* x^{12}, x^{13}, x^{23} such as $x^{13} = (x_{i_1, i_3}^{13}) = (\sum_{i_2} x_{i_1, i_2, i_3})$, and its 1-margins are its *plane-sums* x^1, x^2, x^3 such as $x^2 = (x_{i_2}^2) = (\sum_{i_1, i_3} x_{i_1, i_2, i_3})$.

A *statistical model* is a triple $\mathcal{M} = (d, \mathcal{J}, n)$, where \mathcal{J} is a set of subsets of $\{1, \dots, d\}$ none containing the other and $n = (n_1, \dots, n_d)$ is a tuple of positive integers. The model dictates the collection of margins for d -tables of size n to be specified. The models of primary interest in this article are $(3, \{12, 13, 23\}, (r, c, h))$, that is, 3-tables of size (r, c, h) with all three of their 2-margins specified.

Fix any model $\mathcal{M} = (d, \mathcal{J}, n)$. Any specified collection $u = (u^J : J \in \mathcal{J})$ of margins under this model gives rise to a corresponding set of *contingency tables* with these margins (where \mathbb{N} is the set of nonnegative integers),

$$C(\mathcal{M}; u) := \{x \in \mathbb{N}^{n_1 \times \cdots \times n_d} : x^J = u^J, J \in \mathcal{J}\}.$$

Clearly, this set is precisely the set of integer points in the corresponding transportation polyhedron.

We make several definitions related to the range a table entry can assume. Permuting coordinates if necessary, we may always consider the first entry x_1 , where $\mathbf{1} := (1, \dots, 1)$. The *entry-range* of a collection of margins u under model \mathcal{M} is the set $R(\mathcal{M}; u) := \{x_1 : x \in C(\mathcal{M}; u)\} \subset \mathbb{N}$ of values x_1 can attain in any table with these margins. The *spectrum* of a model \mathcal{M} is the set of all entry-ranges of all collections of margins under \mathcal{M} ,

$$\text{Spec}(\mathcal{M}) := \{R(\mathcal{M}; u) : u = (u^J : J \in \mathcal{J}) \text{ some margin collection under } \mathcal{M}\}.$$

The *spectrum of a class* \mathcal{C} of models is the union $\text{Spec}(\mathcal{C}) := \bigcup_{\mathcal{M} \in \mathcal{C}} \text{Spec}(\mathcal{M})$ of spectra of its models.

The following proposition characterizes the spectra of all-1-margin models and show they always consist precisely of the collection of all intervals, hence “simple”. The proof is not hard and is omitted.

Proposition 1. *Let \mathcal{C}_d^1 be the class of models $(d, \{1, 2, \dots, d\}, (n_1, \dots, n_d))$ of all-1-margins of d -tables. Then for all $d \geq 2$, the spectrum of \mathcal{C}_d^1 is the set of intervals in \mathbb{N} , $\text{Spec}(\mathcal{C}_d^1) = \{[l, u] := \{r \in \mathbb{N} : l \leq r \leq u\} : l, u \in \mathbb{N}\}$.*

An important consequence of our Theorem 1 is the following strikingly surprising result, in contrast with Proposition 1 and with recent attempts by statisticians to better understand entry behavior of slim 3-tables (see [5] and references therein). It says that the class of models of all-2-margin slim 3-tables is *universal* - its spectrum consists of all finite subsets of \mathbb{N} - and hence “complex” and presumably secure.

Corollary 3. *Let \mathcal{S}_3^2 be the class of models $(3, \{12, 13, 23\}, (r, c, 3))$ of all-2-margins of slim 3-tables. Then \mathcal{S}_3^2 is universal, namely, its spectrum $\text{Spec}(\mathcal{S}_3^2)$ is the set of all finite sets of nonnegative integers.*

Proof. Let $S = \{s_1, \dots, s_n\} \subset \mathbb{N}$ be any finite set of nonnegative integers. Consider the polytope

$$P := \{y \in \mathbb{R}_+^{n+1} : y_0 - \sum_{j=1}^n s_j \cdot y_j = 0, \sum_{j=1}^n y_j = 1\}.$$

By Theorem 1, there are r, c and 2-margins $u = (u^{12}, u^{13}, u^{23})$ for the model $\mathcal{M} := (3, \{12, 13, 23\}, (r, c, 3))$, such that the corresponding line-sum slim transportation polytope T represents P . By suitable permutation of coordinates, we can assume that the injection σ giving that representation satisfies $\sigma(0) =$

$(1, 1, 1)$, embedding y_0 as $x_1 = x_{1,1,1}$. The set $C(\mathcal{M}; u)$ of contingency tables is the set of integer points in T , and by Theorem 1, T is integer equivalent to P . The corresponding entry-range is therefore

$$\begin{aligned} R(\mathcal{M}; u) &= \{x_1 : x \in C(\mathcal{M}; u)\} \\ &= \{x_1 : x \in T \cap \mathbb{Z}^{r \times c \times 3}\} = \{y_0 : y \in P \cap \mathbb{Z}^{n+1}\} = S. \end{aligned}$$

So, indeed, any finite $S \subset \mathbb{N}$ satisfies $S = R(\mathcal{M}; u)$ for some margin collection u under some model $\mathcal{M} \in \mathcal{S}_3^2$. \square

We conclude with a (possibly smallest) example of 2-margins of $(6, 4, 3)$ -tables with “complex” entry-range.

Example 3. Gap in entry-range and spectrum of 2-margined 3-tables: we start with the polytope $P = \{y \geq 0 : y_0 - 2y_1 = 0, y_1 + y_2 = 1\}$ in three variables, where the set of integer values y_0 attains is $\{0, 2\}$. Our construction (with a suitable shortcut that gives a smaller array size) represents it as the transportation polytope corresponding to the model $\mathcal{M} = (3, \{12, 13, 23\}, (6, 4, 3))$, with 2-margins $u = (u^{12}, u^{13}, u^{23})$ given by the three matrices below, where the variable y_0 is embedded as the entry $x_{1,1,1}$. By Theorem 1, the corresponding entry-range is $R(\mathcal{M}; u) = \{0, 2\} \in \text{Spec}(\mathcal{M})$ and has a gap.

$$(u_{i_1, i_2}^{12}) = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 \end{pmatrix}, \quad (u_{i_1, i_3}^{13}) = \begin{pmatrix} 2 & 2 & 0 \\ 1 & 1 & 0 \\ 2 & 0 & 2 \\ 3 & 0 & 1 \\ 0 & 2 & 2 \\ 0 & 1 & 3 \end{pmatrix}, \quad (u_{i_2, i_3}^{23}) = \begin{pmatrix} 2 & 3 & 2 \\ 2 & 1 & 2 \\ 2 & 1 & 2 \\ 2 & 1 & 2 \end{pmatrix}.$$

2 The Three-Stage Construction

Our construction consists of three stages which are independent of each other as reflected in Lemma 1 and Theorems 2 and 3 below. Stage one, in §2.1, is a simple preprocessing in which a given polytope is represented as another whose defining system involves only small, $\{-1, 0, 1, 2\}$ -valued, coefficients, at the expense of increasing the number of variables. This enables to make the entire construction run in polynomial-time. However, for systems with small coefficients, such as in the examples above, this may result in unnecessary blow-up and can be skipped. Stage two, in §2.2, represents any rational polytope as a 3-way transportation polytope with specified *plane-sums* and *forbidden-entries*. In the last stage, in §2.3, any plane-sum transportation polytope with upper-bounds on the entries is represented as a slim 3-way line-sum transportation polytope. In §2.4 these three stages are integrated to give Theorem 1, and a complexity estimate is provided.

2.1 Preprocessing: Coefficient Reduction

Let $P = \{y \geq 0 : Ay = b\}$ where $A = (a_{i,j})$ is an integer matrix and b is an integer vector. We represent it as a polytope $Q = \{x \geq 0 : Cx = d\}$, in

polynomial-time, with a $\{-1, 0, 1, 2\}$ -valued matrix $C = (c_{i,j})$ of coefficients, as follows. Consider any variable y_j and let $k_j := \max\{\lfloor \log_2 |a_{i,j}| \rfloor : i = 1, \dots, m\}$ be the maximum number of bits in the binary representation of the absolute value of any $a_{i,j}$. We introduce variables $x_{j,0}, \dots, x_{j,k_j}$, and relate them by the equations $2x_{j,i} - x_{j,i+1} = 0$. The representing injection σ is defined by $\sigma(j) := (j, 0)$, embedding y_j as $x_{j,0}$. Consider any term $a_{i,j} y_j$ of the original system. Using the binary expansion $|a_{i,j}| = \sum_{s=0}^{k_j} t_s 2^s$ with all $t_s \in \{0, 1\}$, we rewrite this term as $\pm \sum_{s=0}^{k_j} t_s x_{j,s}$. To illustrate, consider a system consisting of the single equation $3y_1 - 5y_2 + 2y_3 = 7$. Then the new system is

$$\begin{array}{rcl} 2x_{1,0} - x_{1,1} & = 0 \\ 2x_{2,0} - x_{2,1} & = 0 \\ 2x_{2,1} - x_{2,2} & = 0 \\ & 2x_{3,0} - x_{3,1} = 0 \\ x_{1,0} + x_{1,1} - x_{2,0} & - x_{2,2} & + x_{3,1} = 7 \end{array} .$$

It is easy to see that this procedure provides the sought representation, and we get the following.

Lemma 1. *Any rational polytope $P = \{y \geq 0 : Ay = b\}$ is polynomial-time representable as a polytope $Q = \{x \geq 0 : Cx = d\}$ with $\{-1, 0, 1, 2\}$ -valued defining matrix C .*

2.2 Representing Polytopes as Plane-Sum Entry-Forbidden Transportation Polytopes

Let $P = \{y \geq 0 : Ay = b\}$ where $A = (a_{i,j})$ is an $m \times n$ integer matrix and b is an integer vector: we assume that P is bounded and hence a polytope, with an integer upper bound U (which can be derived from the Cramer's rule bound) on the value of any coordinate y_j of any $y \in P$.

For each variable y_j , let r_j be the largest between the sum of the positive coefficients of y_j over all equations and the sum of absolute values of the negative coefficients of y_j over all equations,

$$r_j := \max \left(\sum_k \{a_{k,j} : a_{k,j} > 0\}, \sum_k \{|a_{k,j}| : a_{k,j} < 0\} \right) .$$

Let $r := \sum_{j=1}^n r_j$, $R := \{1, \dots, r\}$, $h := m + 1$ and $H := \{1, \dots, h\}$. We now describe how to construct vectors $u, v \in \mathbb{Z}^r, w \in \mathbb{Z}^h$, and a set $E \subset R \times R \times H$ of triples - the “enabled”, non-“forbidden” entries - such that the polytope P is represented as the corresponding transportation polytope of $r \times r \times h$ arrays with plane-sums u, v, w and only entries indexed by E enabled,

$$\begin{aligned} T = \{x \in \mathbb{R}_+^{r \times r \times h} : x_{i,j,k} = 0 \text{ for all } (i, j, k) \notin E, \text{ and} \\ \sum_{i,j} x_{i,j,k} = w_k, \sum_{i,k} x_{i,j,k} = v_j, \sum_{j,k} x_{i,j,k} = u_i\} . \end{aligned}$$

We also indicate the injection $\sigma : \{1, \dots, n\} \longrightarrow R \times R \times H$ giving the desired embedding of the coordinates y_j as the coordinates $x_{i,j,k}$ and the representation of P as T (see paragraph following Theorem 1).

Basically, each equation $k = 1, \dots, m$ will be encoded in a “horizontal plane” $R \times R \times \{k\}$ (the last plane $R \times R \times \{h\}$ is included for consistency and its entries can be regarded as “slacks”); and each variable y_j , $j = 1, \dots, n$ will be encoded in a “vertical box” $R_j \times R_j \times H$, where $R = \bigcup_{j=1}^n R_j$ is the natural partition of R with $|R_j| = r_j$, namely with $R_j := \{1 + \sum_{l < j} r_l, \dots, \sum_{l \leq j} r_l\}$.

Now, all “vertical” plane-sums are set to the same value U , that is, $u_j := v_j := U$ for $j = 1, \dots, r$. All entries not in the union $\bigcup_{j=1}^n R_j \times R_j \times H$ of the variable boxes will be forbidden. We now describe the enabled entries in the boxes; for simplicity we discuss the box $R_1 \times R_1 \times H$, the others being similar. We distinguish between the two cases $r_1 = 1$ and $r_1 \geq 2$. In the first case, $R_1 = \{1\}$; the box, which is just the single line $\{1\} \times \{1\} \times H$, will have exactly two enabled entries $(1, 1, k^+), (1, 1, k^-)$ for suitable k^+, k^- to be defined later. We set $\sigma(1) := (1, 1, k^+)$, namely embed $y_1 = x_{1,1,k^+}$. We define the *complement* of the variable y_1 to be $\bar{y}_1 := U - y_1$ (and likewise for the other variables). The vertical sums u, v then force $\bar{y}_1 = U - y_1 = U - x_{1,1,k^+} = x_{1,1,k^-}$, so the complement of y_1 is also embedded. Next, consider the case $r_1 \geq 2$. For each $s = 1, \dots, r_1$, the line $\{s\} \times \{s\} \times H$ (respectively, $\{s\} \times \{1 + (s \bmod r_1)\} \times H$) will contain one enabled entry $(s, s, k^+(s))$ (respectively, $(s, 1 + (s \bmod r_1), k^-(s))$). All other entries of $R_1 \times R_1 \times H$ will be forbidden. Again, we set $\sigma(1) := (1, 1, k^+(1))$, namely embed $y_1 = x_{1,1,k^+(1)}$; it is then not hard to see that, again, the vertical sums u, v force $x_{s,s,k^+(s)} = x_{1,1,k^+(1)} = y_1$ and $x_{s,1+(s \bmod r_1),k^-(s)} = U - x_{1,1,k^+(1)} = \bar{y}_1$ for each $s = 1, \dots, r_1$. Therefore, both y_1 and \bar{y}_1 are each embedded in r_1 distinct entries.

To clarify the above description it may be helpful to visualize the $R \times R$ matrix $(x_{i,j,+})$ whose entries are the vertical line-sums $x_{i,j,+} := \sum_{k=1}^h x_{i,j,k}$. For instance, if we have three variables with $r_1 = 3, r_2 = 1, r_3 = 2$ then $R_1 = \{1, 2, 3\}, R_2 = \{4\}, R_3 = \{5, 6\}$, and the line-sums matrix $x = (x_{i,j,+})$ is

$$\begin{pmatrix} x_{1,1,+} & x_{1,2,+} & 0 & 0 & 0 & 0 \\ 0 & x_{2,2,+} & x_{2,3,+} & 0 & 0 & 0 \\ x_{3,1,+} & 0 & x_{3,3,+} & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{4,4,+} & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{5,5,+} & x_{5,6,+} \\ 0 & 0 & 0 & 0 & x_{6,5,+} & x_{6,6,+} \end{pmatrix} = \begin{pmatrix} y_1 & \bar{y}_1 & 0 & 0 & 0 & 0 \\ 0 & y_1 & \bar{y}_1 & 0 & 0 & 0 \\ \bar{y}_1 & 0 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & U & 0 & 0 \\ 0 & 0 & 0 & 0 & y_3 & \bar{y}_3 \\ 0 & 0 & 0 & 0 & \bar{y}_3 & y_3 \end{pmatrix}.$$

We now encode the equations by defining the horizontal plane-sums w and the indices $k^+(s), k^-(s)$ above as follows. For $k = 1, \dots, m$, consider the k th equation $\sum_j a_{k,j} y_j = b_k$. Define the index sets $J^+ := \{j : a_{k,j} > 0\}$ and $J^- := \{j : a_{k,j} < 0\}$, and set $w_k := b_k + U \cdot \sum_{j \in J^-} |a_{k,j}|$. The last coordinate of w is set for consistency with u, v to be $w_h = w_{m+1} := r \cdot U - \sum_{k=1}^m w_k$. Now, with $\bar{y}_j := U - y_j$ the complement of variable y_j as above, the k th equation can be rewritten as

$$\sum_{j \in J^+} a_{k,j} y_j + \sum_{j \in J^-} |a_{k,j}| \bar{y}_j = \sum_{j=1}^n a_{k,j} y_j + U \cdot \sum_{j \in J^-} |a_{k,j}| = b_k + U \cdot \sum_{j \in J^-} |a_{k,j}| = w_k.$$

To encode this equation, we simply “pull down” to the corresponding k th horizontal plane as many copies of each variable y_j or \bar{y}_j by suitably setting $k^+(s) := k$ or $k^-(s) := k$. By the choice of r_j there are sufficiently many, possibly with a few redundant copies which are absorbed in the last hyperplane by setting $k^+(s) := m+1$ or $k^-(s) := m+1$. For instance, if $m=8$, the first variable y_1 has $r_1 = 3$ as above, its coefficient $a_{4,1} = 3$ in the fourth equation is positive, its coefficient $a_{7,1} = -2$ in the seventh equation is negative, and $a_{k,1} = 0$ for $k \neq 4, 7$, then we set $k^+(1) = k^+(2) = k^+(3) := 4$ (so $\sigma(1) := (1, 1, 4)$ embedding y_1 as $x_{1,1,4}$), $k^-(1) = k^-(2) := 7$, and $k^-(3) := h = 9$.

This way, all equations are suitably encoded, and we obtain the following theorem.

Theorem 2. *Any rational polytope $P = \{y \in \mathbb{R}_+^n : Ay = b\}$ is polynomial-time representable as a plane-sum entry-forbidden 3-way transportation polytope*

$$T = \{x \in \mathbb{R}_+^{r \times r \times h} : x_{i,j,k} = 0 \text{ for all } (i, j, k) \notin E, \text{ and} \\ \sum_{i,j} x_{i,j,k} = w_k, \sum_{i,k} x_{i,j,k} = v_j, \sum_{j,k} x_{i,j,k} = u_i\}.$$

Proof. Follows from the construction outlined above and Lemma 1. \square

2.3 Representing Plane-Sum Entry-Bounded as Slim Line-Sum Entry-Free

Here we start with a transportation polytope of plane-sums and *upper-bounds* $e_{i,j,k}$ on the entries,

$$P = \{y \in \mathbb{R}_+^{l \times m \times n} : \sum_{i,j} y_{i,j,k} = c_k, \sum_{i,k} y_{i,j,k} = b_j, \sum_{j,k} y_{i,j,k} = a_i, y_{i,j,k} \leq e_{i,j,k}\}.$$

Clearly, this is a more general form than that of T appearing in Theorem 2 above - the forbidden entries can be encoded by setting a “forbidding” upper-bound $e_{i,j,k} := 0$ on all forbidden entries $(i, j, k) \notin E$ and an “enabling” upper-bound $e_{i,j,k} := U$ on all enabled entries $(i, j, k) \in E$. Thus, by Theorem 2, any rational polytope is representable also as such a plane-sum entry-bounded transportation polytope P . We now describe how to represent, in turn, such a P as a slim line-sum (entry-free) transportation polytope of the form of Theorem 1,

$$T = \{x \in \mathbb{R}_+^{r \times c \times 3} : \sum_I x_{I,J,K} = w_{J,K}, \sum_J x_{I,J,K} = v_{I,K}, \sum_K x_{I,J,K} = u_{I,J}\}.$$

We give explicit formulas for $u_{I,J}, v_{I,K}, w_{J,K}$ in terms of a_i, b_j, c_k and $e_{i,j,k}$ as follows. Put $r := l \cdot m$ and $c := n + l + m$. The first index I of each entry $x_{I,J,K}$ will be a pair $I = (i, j)$ in the r -set

$$\{(1, 1), \dots, (1, m), (2, 1), \dots, (2, m), \dots, (l, 1), \dots, (l, m)\}.$$

The second index J of each entry $x_{I,J,K}$ will be a pair $J = (s, t)$ in the c -set

$$\{(1, 1), \dots, (1, n), (2, 1), \dots, (2, l), (3, 1), \dots, (3, m)\}.$$

The last index K will simply range in the 3-set $\{1, 2, 3\}$. We represent P as T via the injection σ given explicitly by $\sigma(i, j, k) := ((i, j), (1, k), 1)$, embedding each variable $y_{i,j,k}$ as the entry $x_{(i,j),(1,k),1}$. Let U denote the minimal between the two values $\max\{a_1, \dots, a_l\}$ and $\max\{b_1, \dots, b_m\}$. The line-sums are the matrices

$$(u_{I,J}) = \begin{pmatrix} & 11 & 12 & \cdots & 1n & 21 & 22 & \cdots & 2l & 31 & 32 & \cdots & 3m \\ 11 & e_{1,1,1} & e_{1,1,2} & \cdots & e_{1,1,n} & U & 0 & \cdots & 0 & U & 0 & \cdots & 0 \\ 12 & e_{1,2,1} & e_{1,2,2} & \cdots & e_{1,2,n} & U & 0 & \cdots & 0 & 0 & U & \cdots & 0 \\ \vdots & \vdots \\ 1m & e_{1,m,1} & e_{1,m,2} & \cdots & e_{1,m,n} & U & 0 & \cdots & 0 & 0 & 0 & \cdots & U \\ & & & & & & & & & & & & \\ 21 & e_{2,1,1} & e_{2,1,2} & \cdots & e_{2,1,n} & 0 & U & \cdots & 0 & U & 0 & \cdots & 0 \\ 22 & e_{2,2,1} & e_{2,2,2} & \cdots & e_{2,2,n} & 0 & U & \cdots & 0 & 0 & U & \cdots & 0 \\ \vdots & \vdots \\ 2m & e_{2,m,1} & e_{2,m,2} & \cdots & e_{2,m,n} & 0 & U & \cdots & 0 & 0 & 0 & \cdots & U \\ & & & & & & & & & & & & \\ & & & & & & & & & & & & \\ l1 & e_{l,1,1} & e_{l,1,2} & \cdots & e_{l,1,n} & 0 & 0 & \cdots & U & U & 0 & \cdots & 0 \\ l2 & e_{l,2,1} & e_{l,2,2} & \cdots & e_{l,2,n} & 0 & 0 & \cdots & U & 0 & U & \cdots & 0 \\ \vdots & \vdots \\ lm & e_{l,m,1} & e_{l,m,2} & \cdots & e_{l,m,n} & 0 & 0 & \cdots & U & 0 & 0 & \cdots & U \end{pmatrix}$$

$$(v_{I,K}) = \begin{pmatrix} & 1 & 2 & 3 \\ 11 & U & e_{1,1,+} & U \\ 12 & U & e_{1,2,+} & U \\ \vdots & \vdots & \vdots & \vdots \\ 1m & U & e_{1,m,+} & U \\ & & & & \\ 21 & U & e_{2,1,+} & U \\ 22 & U & e_{2,2,+} & U \\ \vdots & \vdots & \vdots & \vdots \\ 2m & U & e_{2,m,+} & U \end{pmatrix} \quad (w_{J,K}) = \begin{pmatrix} & 1 & 2 & 3 \\ 11 & c_1 & e_{+,+,1} - c_1 & 0 \\ 12 & c_2 & e_{+,+,2} - c_2 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1n & c_n & e_{+,+,n} - c_n & 0 \\ & & & \\ 21 & m \cdot U - a_1 & 0 & a_1 \\ 22 & m \cdot U - a_2 & 0 & a_2 \\ \vdots & \vdots & \vdots & \vdots \\ 2l & m \cdot U - a_l & 0 & a_l \\ & & & \\ 31 & 0 & b_1 & l \cdot U - b_1 \\ 32 & 0 & b_2 & l \cdot U - b_2 \\ \vdots & \vdots & \vdots & \vdots \\ 3m & 0 & b_m & l \cdot U - b_m \end{pmatrix}$$

Theorem 3. *Any rational plane-sum entry-bounded 3-way transportation polytope*

$$P = \{ y \in \mathbb{R}_+^{l \times m \times n} : \sum_{i,j} y_{i,j,k} = c_k, \sum_{i,k} y_{i,j,k} = b_j, \sum_{j,k} y_{i,j,k} = a_i, y_{i,j,k} \leq e_{i,j,k} \}.$$

is strongly-polynomial-time representable as a line-sum slim transportation polytope

$$T = \{ x \in \mathbb{R}_+^{r \times c \times 3} : \sum_I x_{I,J,K} = w_{J,K}, \sum_J x_{I,J,K} = v_{I,K}, \sum_K x_{I,J,K} = u_{I,J} \}.$$

Proof. We outline the proof; complete details are in [4]. First consider any $y = (y_{i,j,k}) \in P$; we claim the embedding via σ of $y_{i,j,k}$ in $x_{(i,j),(1,k),1}$ can be extended uniquely to $x = (x_{I,J,K}) \in T$. First, the entries $x_{I,(3,t),1}$, $x_{I,(2,t),2}$ and $x_{I,(1,t),3}$ for all $I = (i,j)$ and t are zero since so are the line-sums $w_{(3,t),1}$, $w_{(2,t),2}$ and $w_{(1,t),3}$. Next, consider the entries $x_{I,(2,t),1}$: since all entries $x_{I,(3,t),1}$ are zero, examining the line-sums $u_{I,(2,t)}$ and $v_{I,1} = U$, we find $x_{(i,j),(2,i),1} = U - \sum_{t=1}^n x_{(i,j),(1,t),1} = U - y_{i,j,+} \geq 0$ whereas for $t \neq i$ we get $x_{(i,j),(2,t),1} = 0$. This also gives the entries $x_{I,(2,t),3}$: we have $x_{(i,j),(2,i),3} = U - x_{(i,j),(2,i),1} = y_{i,j,+} \geq 0$ whereas for $t \neq i$ we have $x_{(i,j),(2,t),3} = 0$. Next, consider the entries $x_{I,(1,t),2}$: since all entries $x_{I,(1,t),3}$ are zero, examining the line-sums $u_{(i,j),(1,k)} = e_{i,j,k}$ we find $x_{(i,j),(1,k),2} = e_{i,j,k} - y_{i,j,k} \geq 0$ for all i, j, k . Next consider the entries $x_{I,(3,t),2}$: since all entries $x_{I,(2,t),2}$ are zero, examining the line-sums $u_{(i,j),(3,t)}$ and $v_{(i,j),2} = e_{i,j,+}$, we find $x_{(i,j),(3,j),2} = e_{i,j,+} - \sum_{k=1}^l x_{(i,j),(1,k),2} = y_{i,j,+} \geq 0$ whereas for $t \neq j$ we get $x_{(i,j),(3,t),2} = 0$. This also gives the entries $x_{I,(3,t),3}$: we have $x_{(i,j),(3,j),3} = U - x_{(i,j),(3,j),2} = U - y_{i,j,+} \geq 0$ whereas for $t \neq j$ we get $x_{(i,j),(3,t),3} = 0$.

Conversely, given any $x = (x_{I,J,K}) \in T$, let $y = (y_{i,j,k})$ with $y_{i,j,k} := x_{(i,j),(1,k),1}$. Since x is nonnegative, so is y . Further, $e_{i,j,k} - y_{i,j,k} = x_{(i,j),(1,k),2} \geq 0$ for all i, j, k and hence y obeys the entry upper-bounds. Finally, using the relations established above $x_{(i,j),(3,t),2} = 0$ for $t \neq j$, $x_{(i,j),(2,t),3} = 0$ for $t \neq i$, and $x_{(i,j),(3,j),2} = x_{(i,j),(2,i),3} = y_{i,j,+}$, we obtain

$$\sum_{i,j} y_{i,j,k} = \sum_{i,j} x_{(i,j),(1,k),1} = w_{(1,k),1} = c_k, \quad 1 \leq k \leq n;$$

$$\sum_{i,k} y_{i,j,k} = \sum_i x_{(i,j),(3,j),2} = w_{(3,j),2} = b_j, \quad 1 \leq j \leq m;$$

$$\sum_{j,k} y_{i,j,k} = \sum_j x_{(i,j),(2,i),3} = w_{(2,i),3} = a_i, \quad 1 \leq i \leq l.$$

This shows that y satisfies the plane-sums as well and hence is in P . Since integrality is also preserved in both directions, this completes the proof. \square

2.4 The Main Theorem and a Complexity Estimate

Call a class \mathcal{P} of rational polytopes *polynomial-time representable* in a class \mathcal{Q} if there is a polynomial-time algorithm that represents any given $P \in \mathcal{P}$ as some $Q \in \mathcal{Q}$. The resulting binary relation on classes of rational polytopes is clearly transitive. Thus, the composition of Theorem 2.2 (which incorporates Lemma 1) and Theorem 2.3 gives at once Theorem 1 stated in the introduction. Working out the details of our three-stage construction, we can give the following estimate on the number of rows r and columns c in the resulting representing transportation polytope, in terms of the input. The computational complexity of the construction is also determined by this bound, but we do not dwell on the details here.

Theorem 1 (with complexity estimate). *Any polytope $P = \{y \in \mathbb{R}_+^n : Ay = b\}$ with integer $m \times n$ matrix $A = (a_{i,j})$ and integer vector b is polynomial-time representable as a slim transportation polytope*

$$T = \{x \in \mathbb{R}_+^{r \times c \times 3} : \sum_i x_{i,j,k} = w_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\},$$

with $r = O(m^2(n + L)^2)$ rows and $c = O(m(n + L))$ columns, where $L := \sum_{j=1}^n \max_{i=1}^m \lfloor \log_2 |a_{i,j}| \rfloor$.

References

1. Bai  u, M., Balinski, M.L.: The stable allocation (or ordinal transportation) problem. *Math. Oper. Res.* **27** (2002) 485–503
2. Balinski, M.L., Rispoli, F.J.: Signature classes of transportation polytopes. *Math. Prog. Ser. A* **60** (1993) 127–144
3. Cryan, M., Dyer, M., M  ller, H., Stougie, L.: Random walks on the vertices of transportation polytopes with constant number of sources. *Proc. 14th Ann. ACM-SIAM Symp. Disc. Alg.* (Baltimore, MD) 330–339, ACM, New York, 2003
4. De Loera, J., Onn, S.: The complexity of three-way statistical tables. *SIAM J. Comp.*, to appear
5. Duncan, G.T., Fienberg, S.E., Krishnan, R., Padman, R., Roehrig, S.F.: Disclosure limitation methods and information loss for tabular data. In: *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies* (Doyle, P., Land, J.I., Theeuwes, J.M., Zayatz, L.V. eds.), North-Holland, 2001
6. Diaconis, P., Gangolli, A.: Rectangular arrays with fixed margins. In: *Discrete Probability and Algorithms* (Minneapolis, MN, 1993), IMA Vol. Math. App. **72** 15–41, Springer, New York, 1995
7. Fleischer, L.K.: Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Disc. Math.* **13** (2000) 505–520
8. Garg, N., K  nemann, J.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *Proc. 13th Ann. IEEE Symp. Foun. Comp. Sci.* (New York) 300–309, IEEE, New York, 1998
9. Gusfield, D.: A graph theoretic approach to statistical data security. *SIAM J. Comp.* **17** (1988) 552–571

10. Hoşten, S., Sullivant, S.: Gröbner bases and polyhedral geometry of reducible and cyclic models. *J. Combin. Theory Ser. A* **100** (2002) 277–301
11. Hu, T.C.: Multi-commodity network flows. *Oper. Res.* **11** (1963) 344–360
12. Hwang, F.K., Onn, S., Rothblum, U.G.: A polynomial time algorithm for shaped partition problems. *SIAM J. Optim.* **10** (1999) 70–81
13. Irving, R., Jerrum, M.R.: Three-dimensional statistical data security problems. *SIAM J. Comp.* **23** (1994) 170–184
14. Klee, V., Witzgall, C.: Facets and vertices of transportation polytopes. In: *Mathematics of the Decision Sciences, Part I* (Stanford, CA, 1967), 257–282, AMS, Providence, RI, 1968
15. Levin, A.: Personal communication, 2004
16. Mehta, C.R., Patel, N.R.: A network algorithm for performing Fisher's exact test in $r \times c$ contingency tables. *J. Amer. Stat. Assoc.* **78** (1983) 427–434
17. Onn, S., Rothblum, U.G.: Convex combinatorial optimization. *Disc. Comp. Geom.*, to appear
18. Onn, S., Schulman, L.J.: The vector partition problem for convex objective functions. *Math. Oper. Res.* **26** (2001) 583–590
19. Tardos, E.: A strongly polynomial algorithm to solve combinatorial linear programs. *Oper. Res.* **34** (1986) 250–256
20. U.S. Bureau of the Census, *American FactFinder*,
<http://factfinder.census.gov/servlet/BasicFactsServlet>.
21. Vlach, M.: Conditions for the existence of solutions of the three-dimensional planar transportation problem. *Disc. App. Math.*, **13** (1986) 61–78
22. Yemelichev, V.A., Kovalev, M.M., Kravtsov, M.K.: *Polytopes, Graphs and Optimisation*. Cambridge Univ. Press, Cambridge, 1984

A Capacity Scaling Algorithm for M-convex Submodular Flow^{*}

Satoru Iwata¹, Satoko Moriguchi², and Kazuo Murota³

¹ University of Tokyo, Tokyo 113-8656, Japan

² JST, Saitama 332-0012, Japan

³ University of Tokyo, Tokyo 113-8656; PRESTO JST, Saitama, Japan

Abstract. This paper presents a faster algorithm for the M-convex submodular flow problem, which is a generalization of the minimum-cost flow problem with an M-convex cost function for the flow-boundary, where an M-convex function is a nonlinear nonseparable discrete convex function on integer points. The algorithm extends the capacity scaling approach for the submodular flow problem by Fleischer, Iwata and McCormick (2002) with the aid of a novel technique of changing the potential by solving maximum submodular flow problems.

1 Introduction

In recent research towards a unified framework of discrete convex analysis [17], the concept of M-convex functions was proposed by Murota as an extension of that of valuations on matroids invented by Dress and Wenzel [2]. The concept of L-convex functions, which generalize the Lovász extensions of submodular set functions [13], was also proposed. These two concepts of discrete convexity are conjugate to each other, and a Fenchel-type duality theorem holds [16]. This fundamental result in discrete convex analysis is equivalent to the optimality characterization of the M-convex submodular flow problem. In fact, the original proof in [16] of the Fenchel-type duality theorem is based on a cycle-canceling algorithm that solves the M-convex submodular flow problem. In other words, an M-convex submodular flow algorithm naturally provides a method for finding both optima in the min-max relation of the Fenchel-type duality theorem.

The M-convex submodular flow problem is a minimum-cost flow problem with an additional M-convex cost function for the flow-boundary. This is a common generalization of the submodular flow problem of Edmonds and Giles [3] and the valuated matroid intersection problem of Murota [15]. Applications include an economic problem of finding a competitive equilibrium of an economy with indivisible commodities [18]. In this economic problem, consumers' utility functions are assumed to have gross substitutes property, which corresponds to M-concavity. The conjugacy between M-convexity and L-convexity corresponds to the relationship between commodities and prices.

* Supported by the Kayamori Foundation of Informational Science Advancement.

The running time bound of the cycle-canceling algorithm is not polynomial but pseudopolynomial. The first polynomial-time algorithm for the M-convex submodular flow problem is the conjugate scaling algorithm of Iwata and Shigeno [12]. This algorithm extends the cost-scaling primal-dual algorithm due to Cunningham and Frank [1], which is the first polynomial-time combinatorial algorithm for the submodular flow problem. The conjugate scaling algorithm, however, calls a submodular function minimization [11,19] repeatedly, which results in its high cost running time. In fact, the conjugate scaling algorithm using the current fastest submodular function minimization algorithm [10] requires $O(n^8(\log L)^2 \log K)$ evaluations of the M-convex cost function, where n is the number of vertices, L is the maximum absolute value of the capacity, and K is the maximum absolute value of the cost.

On the other hand, the capacity scaling approach of Edmonds and Karp [4] does not admit a straightforward generalization to the submodular flow problem. This is because a naive scaling of a submodular function destroys the submodularity. The first capacity scaling algorithm for the submodular flow problem was obtained by Iwata [9] with a subsequent improvement in time complexity by Fleischer, Iwata and McCormick [5]. These algorithms make use of additional arcs, called relaxation arcs, to retain submodularity under the scaling.

This paper aims at extending the capacity scaling approach to obtain an efficient algorithm for the M-convex submodular flow problem under the assumption that an oracle for computing M-convex function values is available. An earlier attempt in this direction by Moriguchi and Murota [14] led to a capacity-scaling successive-shortest-path algorithm applicable only to a subclass of M-convex functions. This paper extends the relaxation arc technique by introducing a new procedure that updates the potential by solving a maximum submodular flow problem. The resulting algorithm provides an optimal solution under the assumption that an oracle for computing M-convex function values is available. The number of oracle calls is bounded by $O(n^6(\log L)^2 \log(nK))$. This is not only the fastest known algorithm but also the first combinatorial polynomial-time algorithm that solves the M-convex submodular flow problem without relying on submodular function minimization.

2 M-convex Submodular Flow Problem

Let V be a finite set of cardinality n , and \mathbf{Z} be the set of integers. Let $\chi_v \in \{0, 1\}^V$ denote the characteristic vector of $v \in V$. The characteristic vector of $X \subseteq V$ is denoted by χ_X . For $x \in \mathbf{Z}^V$ and $v \in V$, $x(v)$ means the component of x with index v . For $x \in \mathbf{Z}^V$ and $X \subseteq V$, we write $x(X) = \sum_{v \in X} x(v)$. We write the positive and negative supports of a vector x by $\text{supp}^+(x) = \{v \in V \mid x(v) > 0\}$, $\text{supp}^-(x) = \{v \in V \mid x(v) < 0\}$. For a function $f : \mathbf{Z}^V \rightarrow \mathbf{Z} \cup \{+\infty\}$, we use the notation $\text{dom } f = \{x \in \mathbf{Z}^V \mid f(x) < +\infty\}$ for the effective domain of f .

A function $f : \mathbf{Z}^V \rightarrow \mathbf{Z} \cup \{+\infty\}$ is said to be M-convex if it satisfies

(M-EXC) $\forall x, y \in \text{dom } f, \forall u \in \text{supp}^+(x - y), \exists v \in \text{supp}^-(x - y)$ such that $f(x) + f(y) \geq f(x - \chi_u + \chi_v) + f(y + \chi_u - \chi_v)$.

Throughout this paper, let f be an M-convex function $f : \mathbf{Z}^V \rightarrow \mathbf{Z} \cup \{+\infty\}$. It follows from (M-EXC) that $B = \text{dom } f$ satisfies the following property:

(B-EXC) $\forall x, y \in B, \forall u \in \text{supp}^+(x - y), \exists v \in \text{supp}^-(x - y)$ such that $x - \chi_u + \chi_v \in B, y + \chi_u - \chi_v \in B$.

Note that (B-EXC) implies $x(V) = y(V)$ for any $x, y \in B$. A nonempty set $B \subseteq \mathbf{Z}^V$ with the property (B-EXC) is called an M-convex set.

A set function $\rho : 2^V \rightarrow \mathbf{Z}$ is said to be submodular if it satisfies

$$\rho(X) + \rho(Y) \geq \rho(X \cup Y) + \rho(X \cap Y) \quad (X, Y \subseteq V).$$

With a submodular function ρ , we associate the base polyhedron $\mathbf{B}(\rho)$ defined by

$$\mathbf{B}(\rho) = \{x \in \mathbf{R}^V \mid x(V) = \rho(V); \forall X \subseteq V : x(X) \leq \rho(X)\}.$$

A bounded set $B \subseteq \mathbf{Z}^V$ is M-convex if and only if $B = \mathbf{B}(\rho) \cap \mathbf{Z}^V$ for some submodular function $\rho : 2^V \rightarrow \mathbf{Z}$. Accordingly, a member of an M-convex set is referred to as a base. For any base $x \in B$ and $u, v \in V$ with $u \neq v$, the integer defined by $\sigma(x, v, u) = \max\{\alpha \in \mathbf{Z} \mid x - \alpha(\chi_u - \chi_v) \in B\}$ is called the exchange capacity.

Let $G = (V, A)$ be a directed graph with lower and upper capacity bounds $b, c : A \rightarrow \mathbf{Z}$ and the cost function $d : A \rightarrow \mathbf{Z}$. For each vertex $v \in V$, let δ^+v (resp., δ^-v) denote the set of arcs leaving (resp., entering) v . For each vertex subset $X \subseteq V$, let Δ^+X (resp., Δ^-X) denote the set of arcs leaving (resp., entering) X . For each arc $a \in A$, ∂^+a designates the initial vertex of a , and ∂^-a the terminal vertex of a . The boundary $\partial\xi$ of flow $\xi : A \rightarrow \mathbf{Z}$, which represents the net flow leaving vertex v , is defined as $\partial\xi(v) = \sum_{a \in \delta^+v} \xi(a) - \sum_{a \in \delta^-v} \xi(a)$ for each $v \in V$.

The M-convex submodular flow problem (MCSFP) is a minimum-cost flow problem with an M-convex cost function for the flow boundary $\partial\xi$.

$$\text{MCSFP: Minimize} \quad \sum_{a \in A} d(a)\xi(a) + f(\partial\xi) \quad (1)$$

$$\text{subject to} \quad b(a) \leq \xi(a) \leq c(a) \quad (a \in A), \quad (2)$$

$$\partial\xi \in \text{dom } f, \quad (3)$$

$$\xi(a) \in \mathbf{Z} \quad (a \in A). \quad (4)$$

If f is $\{0, +\infty\}$ -valued, then the MCSFP is the submodular flow problem. For the sake of simplicity we assume that $\text{dom } f$ is bounded, which implies the existence of a submodular function $\rho : 2^V \rightarrow \mathbf{Z}$ such that $\text{dom } f = \mathbf{B}(\rho) \cap \mathbf{Z}^V$. A feasible flow for an MCSFP means a function $\xi : A \rightarrow \mathbf{Z}$ that satisfies (2) and (3). An MCSFP is called feasible if it admits a feasible flow.

Theorem 1 ([7]). *An M-convex submodular flow problem MCSFP for a bounded effective domain $\text{dom } f$ is feasible if and only if $b(\Delta^+X) - c(\Delta^-X) \leq \rho(X)$ for any $X \subseteq V$, where ρ is the submodular set function that determines $\text{dom } f$ by $\text{dom } f = \mathbf{B}(\rho) \cap \mathbf{Z}^V$.*

The feasibility of an MCSFP can be checked efficiently [6,8], provided that an initial base $x \in \text{dom } f$ is available. In the rest of this paper, we assume that the given MCSFP is feasible. This assumption implies that

$$\text{dom } f \subseteq \{x \in \mathbf{Z}^V \mid x(V) = 0\}. \quad (5)$$

For f and a vector $p = (p(v) \mid v \in V)$, we define a function $f[-p]$ as $f[-p](x) = f(x) - \langle p, x \rangle$ for each $x \in \mathbf{Z}^V$, where $\langle p, x \rangle = \sum_{v \in V} p(v)x(v)$. This is M-convex for M-convex f .

Lemma 1 ([17]). *For each $p \in \mathbf{Z}^V$, $\arg \min f[-p]$ is an M-convex set.*

By a potential we mean a vector $p \in \mathbf{Z}^V$. The reduced cost $d_p : A \rightarrow \mathbf{Z}$ is defined by $d_p(a) = d(a) + p(\partial^+ a) - p(\partial^- a)$ for each $a \in A$.

Theorem 2 ([16,17]). *A feasible flow ξ is optimal if and only if there exists a potential $p \in \mathbf{Z}^V$ and a base $x \in \text{dom } f$ such that*

- (A) $d_p(a) > 0 \implies \xi(a) = b(a)$ and $d_p(a) < 0 \implies \xi(a) = c(a)$,
- (B) $x \in \arg \min f[-p]$,
- (C) $x = \partial \xi$.

The conjugate function $g : \mathbf{Z}^V \rightarrow \mathbf{Z} \cup \{+\infty\}$ of an M-convex f , defined as

$$g(p) = \sup\{\langle p, x \rangle - f(x) \mid x \in \mathbf{Z}^V\},$$

is another kind of discrete convex function, called an L-convex function. For vectors $p, q \in \mathbf{Z}^V$, we write $p \vee q$ and $p \wedge q$ for their componentwise maximum and minimum. We write $\mathbf{1} = (1, 1, \dots, 1) \in \mathbf{Z}^V$. A function $g : \mathbf{Z}^V \rightarrow \mathbf{Z} \cup \{+\infty\}$ is called L-convex if it satisfies

- (SBF) $g(p) + g(q) \geq g(p \vee q) + g(p \wedge q) \quad (p, q \in \mathbf{Z}^V)$,
- (TRF) $\exists r \in \mathbf{Z}$ such that $g(p + \mathbf{1}) = g(p) + r \quad (p \in \mathbf{Z}^V)$.

An L-convex function g satisfies the discrete midpoint convexity

$$g(p) + g(q) \geq g\left(\left\lceil \frac{p+q}{2} \right\rceil\right) + g\left(\left\lfloor \frac{p+q}{2} \right\rfloor\right) \quad (p, q \in \mathbf{Z}^V), \quad (6)$$

where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ mean the rounding-up and -down of a vector to the nearest integer vector.

An alternative representation of $\arg \min f[-p]$ in condition (B) of Theorem 2 is obtained as follows. Let g be the conjugate of the M-convex function f . Since $\text{dom } f$ is bounded, g is finite-valued. Furthermore, (5) implies that $g(p + \mathbf{1}) = g(p)$ for any $p \in \mathbf{Z}^V$.

Lemma 2 ([17]). *Let $g_p : 2^V \rightarrow \mathbf{Z}$ be the submodular set function defined by $g_p(X) = g(p + \chi_X) - g(p)$. Then, we have $\arg \min f[-p] = \mathbf{B}(g_p) \cap \mathbf{Z}^V$.*

3 A Capacity Scaling Algorithm

3.1 Algorithm Outline

Our capacity scaling algorithm for MCSFP is an improvement of the successive shortest path (SSP) algorithm proposed in [14]. The running time bound is polynomial in n , $\log L$, and $\log K$, where

$$L = \max\{\max_{a \in A} |b(a)|, \max_{a \in A} |c(a)|, \max_{x, y \in \text{dom } f} \|x - y\|_\infty\},$$

$$K = \max\{\max_{a \in A} |d(a)|, \max_{x, y \in \text{dom } f} |f(x) - f(y)|\}.$$

Note that K and L are used as measures of the problem size. We do not need to evaluate their values in our algorithm.

The algorithm performs a number of scaling phases for different values of a scaling parameter α that represents a scale for the arc capacity and $\text{dom } f$. A scaling phase with a specific value of α is called the α -scaling phase. The scaling parameter α is initially set to a sufficiently large number and reduced by a factor of two in each scaling phase. At the end of the α -scaling phase, the algorithm obtains a flow and a potential that satisfy the optimality criterion in Theorem 2 only approximately, with precision α , so to speak. We refer to this condition as α -optimality, of which we give the precise definition later. At the final stage, after the execution of the 1-scaling phase, the algorithm calls algorithm SSP to end up with an optimal flow. We describe SSP in Fig. 1.

Algorithm: SSP(x, ξ, p) Output: An optimal flow ξ for MCSFP. begin while $S^+ \neq \emptyset$ do begin compute the shortest path distance $q(v)$ from S^+ to each $v \in V \setminus S^+$ in $G(\xi, x)$ with respect to the reduced length l_p ; let P be the one with the minimum number of arcs among the shortest paths from S^+ to S^- ; for $v \in V$ do $p(v) := p(v) + \min\{q(v), \sum_{a \in P} l_p(a)\}$; for $a \in F_\xi \cap P$ do $\xi(a) := \xi(a) + 1$; for $a \in B_\xi \cap P$ do $\xi(\bar{a}) := \xi(\bar{a}) - 1$; for $a \in E_x \cap P$ do $x(\partial^+ a) := x(\partial^+ a) - 1, x(\partial^- a) := x(\partial^- a) + 1$ end; return ξ end
--

Fig. 1. Successive shortest path algorithm

Just as the SSP algorithm, the capacity scaling algorithm maintains a flow $\xi : A \rightarrow \mathbf{Z}$, a vector $x \in \mathbf{Z}^V$ and a potential $p \in \mathbf{Z}^V$ and works with an auxiliary

graph. The capacity constraint (2) for the flow ξ is always retained. The flow boundary $\partial\xi$ may differ from the vector x , but their discrepancy is controlled by the scaling parameter α so that $\|x - \partial\xi\|_1$ becomes sufficiently small, i.e., at most $2\alpha n^2$, at the end of the α -scaling phase.

To incorporate the scaling approach into the SSP framework, our algorithm adds a complete directed graph on V with arc set $D = \{(u, v) \mid u \neq v \in V\}$ to the given graph (V, A) , and considers a flow $\psi : D \rightarrow \mathbf{Z}$ on it. The added arcs are called the relaxation arcs and the flow ψ is called the relaxation flow. The relaxation arcs have capacity α , so that

$$0 \leq \psi(a) \leq \alpha \quad (a \in D). \quad (7)$$

Accordingly, the capacity functions b and c are extended to D as $b(a) = 0$ and $c(a) = \alpha$ for all $a \in D$. The algorithm always retains this capacity constraints for ψ . We impose that, for any distinct $u, v \in V$, at least one of $\psi(u, v)$ and $\psi(v, u)$ should be zero. Thus the algorithm maintains a pair of flows ξ on A and ψ on D . In accordance with the introduction of the relaxation flow ψ , the algorithm uses another vector y to represent $x + \partial\psi$, i.e., it always keeps the relationship

$$y = x + \partial\psi. \quad (8)$$

In the algorithm, we always keep y as a base in $\text{dom } f$.

We now define α -optimality, which is a relaxation of the optimality criterion in Theorem 2. A triple (ξ, x, ψ) is said to be α -optimal if it satisfies

- (A[α]) $d_p(a) > 0 \implies b(a) \leq \xi(a) < b(a) + \alpha,$
 $d_p(a) < 0 \implies c(a) \geq \xi(a) > c(a) - \alpha,$
- (B[α]) $y = x + \partial\psi \in \arg \min f[-p],$
- (C[α]) $\|x - \partial\xi\|_1 \leq 2\alpha n^2.$

Each scaling phase of our algorithm aims at finding an α -optimal (ξ, x, ψ) . Once the algorithm obtains an α -optimal (ξ, x, ψ) , it cuts the value of α in half and goes to the next phase. The relationship among the variables in each α -scaling phase is summarized in Fig. 2.

The auxiliary graph is defined with reference to a pair of flows, ξ and ψ , and a base y (not x), as follows. The vertex set is V and the arc set consists of three disjoint parts, A_ξ , E_y , and D_ψ ; i.e., $G(\xi, y, \psi) = (V, A_\xi \cup E_y \cup D_\psi)$. The set A_ξ is given by $A_\xi = F_\xi \cup B_\xi$ with

$$\begin{aligned} F_\xi &= \{a \mid a \in A, \xi(a) < c(a)\}, \\ B_\xi &= \{\bar{a} \mid a \in A, b(a) < \xi(a)\} \quad (\bar{a}: \text{reorientation of } a), \end{aligned}$$

whereas D_ψ is given by $D_\psi = \{a \mid a \in D, \psi(a) < \alpha\}$, and E_y is the arc set of the exchangeability graph for $y \in \text{dom } f$.

On the arc set $A_\xi \cup D_\psi$, we define a function $r : A_\xi \cup D_\psi \rightarrow \mathbf{Z}$, representing residual capacities, as

$$r(a) = \begin{cases} c(a) - \xi(a) & (a \in F_\xi), \\ \xi(\bar{a}) - b(\bar{a}) & (a \in B_\xi, \bar{a} \in A), \\ \alpha - \psi(a) & (a \in D_\psi). \end{cases}$$

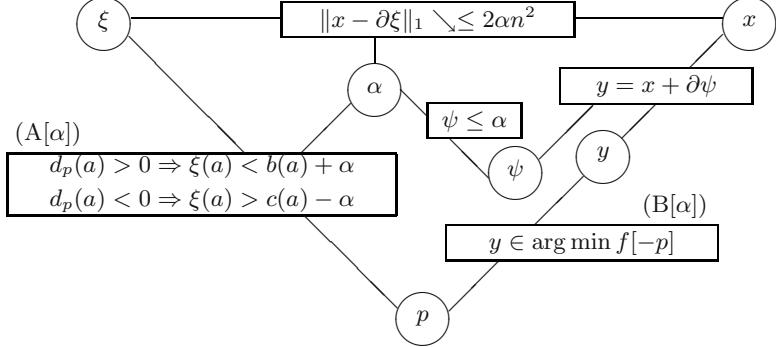


Fig. 2. Relationship among variables in a scaling phase

We define the sets of arcs with residual capacities at least α as

$$\begin{aligned} F_\xi(\alpha) &= \{a \mid a \in F_\xi, r(a) \geq \alpha\}, \quad B_\xi(\alpha) = \{a \mid a \in B_\xi, r(a) \geq \alpha\}, \\ A_\xi(\alpha) &= \{a \mid a \in A_\xi, r(a) \geq \alpha\}, \quad D_\psi(\alpha) = \{a \mid a \in D_\psi, \psi(a) = 0\}. \end{aligned}$$

Furthermore, on the arc set $A_\xi \cup E_y$, we define a function $l : A_\xi \cup E_y \rightarrow \mathbf{Z}$, representing arc lengths, as

$$l(a) = \begin{cases} d(a) & (a \in F_\xi), \\ -d(\bar{a}) & (a \in B_\xi, \bar{a} \in A), \\ \Delta f(y; v, u) & (a = (u, v) \in E_y). \end{cases}$$

Given a potential $p \in \mathbf{Z}^V$, we define the reduced length $l_p : A_\xi \cup E_y \rightarrow \mathbf{Z}$ with respect to p as $l_p(a) = l(a) + p(\partial^+ a) - p(\partial^- a)$ for $a \in A_\xi \cup E_y$. In terms of this reduced length, our conditions (A[α]) and (B[α]) can be put in a more compact form, i.e., $l_p(a) \geq 0$ for $a \in A_\xi(\alpha) \cup E_y$. The algorithm always maintains this nonnegativity of the reduced length for the arcs in $A_\xi(\alpha) \cup E_y$.

We now intend to reduce the discrepancy between x and $\partial\xi$ by augmenting α units of flow along a path P from $S^+(\alpha) = \{v \in V \mid x(v) - \partial\xi(v) \geq \alpha\}$ to $S^-(\alpha) = \{v \in V \mid x(v) - \partial\xi(v) \leq -\alpha\}$, while maintaining (A[α]) and (B[α]). Such a path that allows α -units of flow augmentation should consist of arcs in $A_\xi(\alpha) \cup D_\psi(\alpha)$. In order to satisfy (A[α]) after the augmentation, we further impose that $l_p(a) = 0$ on arcs $a \in A_\xi(\alpha) \cap P$. That is, the path P consists of arcs in $A^* \cup D^*$, where $A^* = F^* \cup B^*$ with

$$F^* = \{a \mid a \in F_\xi(\alpha), l_p(a) = 0\}, \quad B^* = \{a \mid a \in B_\xi(\alpha), l_p(a) = 0\},$$

and $D^* = D_\psi(\alpha)$. We call such a path an α -augmenting path. In Section 3.2, we describe a path search procedure that finds an α -augmenting path by possibly changing ψ , y and p while maintaining (B[α]).

We augment the flow along an α -augmenting path P . The flow augmentation along P means that we update $\xi(a)$ and $\psi(a)$ for each arc $a \in P$ as follows.

- If $a \in F^*$, then $\xi(a) := \xi(a) + \alpha$.
- If $a \in B^*$, then $\xi(\bar{a}) := \xi(\bar{a}) - \alpha$.
- If $a \in D^*$, then $\psi(a) := \alpha - \psi(\bar{a})$, $\psi(\bar{a}) := 0$.

Compute $x = y - \partial\psi$. This flow augmentation decreases $\|y - (\partial\xi + \partial\psi)\|_1$ at both endpoints of the path P and does not change at the other vertices. Since $x - \partial\xi = y - (\partial\xi + \partial\psi)$, $\|x - \partial\xi\|_1$ is reduced by 2α . The flow augmentation does not violate $(B[\alpha])$ as y and p are unchanged. The condition $(A[\alpha])$ is also preserved since $l_p(a) = 0$ for arcs $a \in A_\xi(\alpha) \cap P$ and an arc with residual capacity newly at least α is the reverse arc of an arc $a \in P$.

The algorithm repeats the path search and flow augmentation until we obtain an α -optimal (ξ, x, ψ) . Then we go to the next scaling phase by cutting the value of α in half.

At the start of a new α -scaling phase, we modify ψ as $\psi(a) := \min\{\psi(a), \alpha\}$ for $a \in D$ to satisfy the capacity constraints (7) for the new value of α , and modify x to $y - \partial\psi$ to maintain (8). We also modify ξ as follows to resolve the possible violation of $(A[\alpha])$.

- If $a \in F_\xi(\alpha)$ and $l_p(a) < 0$, then $\xi(a) := \xi(a) + \alpha$.
- If $a \in B_\xi(\alpha)$ and $l_p(a) < 0$, then $\xi(\bar{a}) := \xi(\bar{a}) - \alpha$.

The algorithm also updates the auxiliary graph $G(\xi, y, \psi)$ after this adjustment.

As an initial flow, we adopt any ξ that satisfies (2) and (A) in Theorem 2 for $p = \mathbf{0}$. We also adopt any $x \in \arg \min f$ as an initial base. For these ξ and x , the initial value of α is set to be $2^{\lfloor \log M \rfloor}$, where $M = \|x - \partial\xi\|_\infty/n$. Since we assume MCSFP is feasible, we have a feasible flow ξ° , which satisfies $\|x - \partial\xi^\circ\|_\infty \leq L$ and $\|\partial\xi^\circ - \partial\xi\|_\infty \leq 2(n-1)L$. This implies $M = \|x - \partial\xi\|_\infty/n \leq 2L$. We initialize $p = \mathbf{0}, \psi = \mathbf{0}$ and $y = x$.

The algorithm CAPACITY-SCALING is described in Fig. 3.

3.2 Path Search

In each phase of CAPACITY-SCALING, we find an α -augmenting path by calling procedure PATH-SEARCH of Fig. 4 repeatedly.

At the start of PATH-SEARCH, we call procedure POTENTIAL-UPDATE to find a potential p which satisfies

$$|p(u) - p(v)| \leq (n-1)K \quad (u, v \in V) \tag{9}$$

in addition to $(A[\alpha])$ and $(B[\alpha])$ for the current ξ and y . Let $G_\alpha(\xi, y)$ be a graph with the vertex set V and the arc set $A_\xi(\alpha) \cup E_y$, i.e., $G_\alpha(\xi, y) = (V, A_\xi(\alpha) \cup E_y)$. At the start of POTENTIAL-UPDATE, we identify the strongly connected components of $G_\alpha(\xi, y)$ and its source components, i.e., components without entering arcs. We choose a vertex v_C from each source component C . Let S be

```

Algorithm: CAPACITY_SCALING
Output: An optimal flow  $\xi$  for MCSFP.

begin
set  $p := \mathbf{0}$ ;
find  $\xi$  which satisfies (A) and (2);
find  $x \in \arg \min f$ ;
set  $\psi := \mathbf{0}$  and  $y := x$ ;
set  $M := \|x - \partial\xi\|_\infty/n$  and  $\alpha := 2^{\lfloor \log M \rfloor}$ ;
while  $\alpha \geq 1$  do
begin
for  $a \in D$  do  $\psi(a) := \min\{\psi(a), \alpha\}$ ;
 $x := y - \partial\psi$ ;
for  $a \in F_\xi(\alpha)$  do if  $l_p(a) < 0$  then  $\xi(a) := \xi(a) + \alpha$ ;
for  $a \in B_\xi(\alpha)$  do if  $l_p(a) < 0$  then  $\xi(\bar{a}) := \xi(\bar{a}) - \alpha$ ;
while  $\|x - \partial\xi\|_1 > 2\alpha n^2$  do
begin
 $P := \text{PATH\_SEARCH}(\xi, y, \psi, p, \alpha)$ ; [This updates  $\psi, y, p$ .]
for  $a \in F^* \cap P$  do  $\xi(a) := \xi(a) + \alpha$ ;
for  $a \in B^* \cap P$  do  $\xi(\bar{a}) := \xi(\bar{a}) - \alpha$ ;
for  $a \in D^* \cap P$  do  $\psi(a) := \alpha - \psi(\bar{a}), \psi(\bar{a}) := 0$ ;
 $x := y - \partial\psi$ 
end;
 $\alpha := \alpha/2$ 
end;
 $\xi := \text{SSP}(y, \xi, p)$ ;
return  $\xi$ 
end

```

Fig. 3. Capacity scaling algorithm

the collection of v_C for all source components. We compute the shortest path distance $q(v)$ from S to each $v \in V$ in $G_\alpha(\xi, y)$ with respect to the arc length l . We adopt this q as the new potential p .

Procedure PATH_SEARCH is a variant of the ordinary graph search on the graph $(V, A^* \cup D^*)$. Let R denote the set of the vertices reached from $S^+(\alpha)$ at a certain point in the search, and put $W = V \setminus R$. If there exists an arc in $A^* \cup D^*$ from R to W , PATH_SEARCH extends the search tree in the usual manner. Otherwise, in order to create a new arc that can be used to extend the search tree while maintaining both conditions (A[α]) and (B[α]), it modifies p , y and ψ to $p' = p + \pi\chi_W$, $y' = y - \partial\varphi$, and $\psi' = \psi - \varphi$ with a nonnegative integer π and an integer-flow φ on $\vec{D} = \{(u, v) \mid (u, v) \in D, u \in R, v \in W\}$ determined by procedure CUT_CANCEL. During an execution of CUT_CANCEL and after the above modifications of p , y and ψ , we keep $y - \partial\psi$ unchanged.

We now discuss how to determine such an appropriate pair of π and φ that yields a new arc in $A^* \cup D^*$. We impose the condition $0 \leq \varphi(a) \leq \psi(a)$ for $a \in \vec{D}$ to maintain (7) for ψ' . We also impose $y - \partial\varphi \in \arg \min f[-p - \pi\chi_W]$ to maintain (B[α]) for (y', p') . For the remaining condition (A[α]), we impose a

```

Procedure: PATH_SEARCH( $\xi, y, \psi, p, \alpha$ )
Output: An  $\alpha$ -augmenting path  $P$  from  $S^+(\alpha)$  to  $S^-(\alpha)$ .
begin
 $p :=$  POTENTIAL_UPDATE( $\xi, y, \alpha$ );
 $Q := S^+(\alpha)$ ,  $R := S^+(\alpha)$ ,  $T := \emptyset$ ;
while  $R \cap S^-(\alpha) = \emptyset$  do
begin
if  $Q \neq \emptyset$  then
begin
choose  $u \in Q$ ;
if there exists  $v \in V \setminus R$  with  $a = (u, v) \in A^* \cup D^*$  then
 $R := R \cup \{v\}$ ,  $Q := Q \cup \{v\}$ ,  $T := T \cup \{a\}$ 
else  $Q := Q \setminus \{u\}$ 
end
else
begin
 $(\pi, \varphi) :=$  CUT_CANCEL( $\xi, y, \psi, p, \alpha$ );
for  $v \in V \setminus R$  do  $p(v) := p(v) + \pi$ ;
for  $v \in V$  do  $y(v) := y(v) - \partial\varphi(v)$ ;
for  $a \in \vec{D}$  do  $\psi(a) := \psi(a) - \varphi(a)$ ;
 $Q := R$ 
end
end;
return a path  $P$  from  $S^+(\alpha)$  to  $S^-(\alpha)$  using vertices in  $R$  and arcs in  $T$ 
end

```

Fig. 4. Procedure PATH_SEARCH

further condition $\pi \leq l_p(a)$ for each arc a in $\vec{A}_\xi(\alpha) = \{a \mid a \in A_\xi(\alpha), \partial^+a \in R, \partial^-a \in W\}$. To sum up, CUT_CANCEL aims at finding (π, φ) such that

$$0 \leq \pi \leq l_p(a) \quad (a \in \vec{A}_\xi(\alpha)), \quad (10)$$

$$0 \leq \varphi(a) \leq \psi(a) \quad (a \in \vec{D}), \quad (11)$$

$$y - \partial\varphi \in \arg \min f[-p - \pi \chi_W], \quad (12)$$

and that $D_{\psi'}(\alpha) \cup \{a \mid a \in \vec{A}_\xi(\alpha), l_{p'}(a) = 0\}$ contains an arc from R to W .

It turns out that such (π, φ) exists and can be determined with the aid of the following maximum submodular flow problem on a bipartite graph $(R, W; \vec{D})$.

$$\text{MaxSFP}(\pi): \text{ Maximize } \sum_{a \in \vec{D}} \varphi(a) \quad (13)$$

$$\text{subject to } 0 \leq \varphi(a) \leq \psi(a) \quad (a \in \vec{D}), \quad (14)$$

$$y - \partial\varphi \in \arg \min f[-p - \pi \chi_W], \quad (15)$$

$$\varphi(a) \in \mathbf{Z} \quad (a \in \vec{D}). \quad (16)$$

It should be clear that $\varphi : \vec{D} \rightarrow \mathbf{Z}$ is the variable to be optimized and π is a parameter that specifies the problem. Since $\arg \min f[-p - \pi \chi_W]$ is an M-convex

set by Lemma 1, this problem is a maximum integral submodular flow problem. For the maximum submodular flow problem, a number of efficient algorithms are available [6,8].

If $\vec{A}_\xi(\alpha)$ is nonempty, CUT-CANCEL computes

$$\bar{\pi} = \min\{l_p(a) \mid a \in \vec{A}_\xi(\alpha)\}. \quad (17)$$

If MaxSFP($\bar{\pi}$) is feasible, then it returns $\bar{\pi}$ and an optimal flow $\bar{\varphi}$ of MaxSFP($\bar{\pi}$). Consequently, $l_{p'}(a) = 0$ holds for the arc $a \in \vec{A}_\xi(\alpha)$ that attains the minimum on the right-hand side of (17).

If MaxSFP($\bar{\pi}$) is infeasible or $\vec{A}_\xi(\alpha)$ is empty, CUT-CANCEL looks for π^* such that MaxSFP(π^*) is feasible and MaxSFP($\pi^* + 1$) is infeasible. If $\vec{A}_\xi(\alpha)$ is nonempty and MaxSFP($\bar{\pi}$) is infeasible, such π^* exists between 0 and $\bar{\pi}$ because MaxSFP(0) is feasible. We adopt $\pi^\circ = 0$ and $\pi^\bullet = \bar{\pi}$ as lower and upper bounds on π^* . If $A_\xi(\alpha)$ is empty, then we check the feasibility of MaxSFP(π) for $\pi = 1, 2, 4, 8, \dots$, until we reach infeasible MaxSFP(π). The last π serves as an upper bound π^\bullet and the previous π as a lower bound π° on π^* . It will be shown later in Lemma 5 that π^\bullet is at most $2nK$.

We then find π^* by the following ordinary bisection procedure starting with lower and upper bounds π° and π^\bullet . At every iteration, check the feasibility of MaxSFP(π) for $\pi = \lceil(\pi^\circ + \pi^\bullet)/2\rceil$. If it is infeasible, then reset π^\bullet to $\lceil(\pi^\circ + \pi^\bullet)/2\rceil$. Otherwise, reset π° to $\lceil(\pi^\circ + \pi^\bullet)/2\rceil$. Repeat this process until $\pi^\bullet = \pi^\circ + 1$. Then π° gives the desired value of π^* . We return π^* and an optimal flow φ^* of MaxSFP(π^*). It will be shown later in Lemma 4 that we have $\vec{D} \cap D_{\psi'}(\alpha) \neq \emptyset$ for $\psi' = \psi - \varphi^*$.

We summarize the procedure CUT-CANCEL in Fig. 5.

As a result of CUT-CANCEL, an arc is added to the search tree and the set R becomes larger. This implies the following lemma.

Lemma 3. *In one execution of PATH-SEARCH, CUT-CANCEL is called at most $n - 1$ times.*

For the justification of PATH-SEARCH, it now remains to show that $\vec{D} \cap D_{\psi'}(\alpha) \neq \emptyset$ for $\psi' = \psi - \varphi^*$ and that $\pi^\bullet \leq 2nK$.

Lemma 4. *Suppose that φ^* is optimal to MaxSFP(π^*) and that MaxSFP($\pi^* + 1$) is infeasible. Then there exists an arc $a \in \vec{D}$ such that $\varphi^*(a) = \psi(a)$, and hence $\vec{D} \cap D_{\psi'}(\alpha) \neq \emptyset$ for $\psi' = \psi - \varphi^*$.*

Proof. It suffices to show that, if φ is an optimal flow of MaxSFP(π) and $\varphi(a) < \psi(a)$ for every arc $a \in \vec{D}$, then φ is also feasible for MaxSFP($\pi + 1$). Define $\rho_\pi(X) = g(p + \pi\chi_W + \chi_X) - g(p + \pi\chi_W)$, which is the submodular function associated with the M-convex set $\arg \min f[-p - \pi\chi_W]$; see Lemma 2. For $z = y - \partial\varphi \in \arg \min f[-p - \pi\chi_W]$, we have

$$z(X) = y(X) - \partial\varphi(X) \leq \rho_\pi(X) \quad (X \subseteq V). \quad (18)$$

```

Procedure: CUT_CANCEL( $\xi, y, \psi, p, \alpha$ )
Output: A pair of  $\pi$  and  $\varphi$  that satisfy (10)–(12).
begin
  if  $\vec{A}_\xi(\alpha) \neq \emptyset$  then
    begin
       $\bar{\pi} := \min\{l_p(a) \mid a \in \vec{A}_\xi(\alpha)\}$ ;
      if MaxSFP( $\bar{\pi}$ ) is feasible then
        begin
           $\bar{\varphi} :=$  optimal flow for MaxSFP( $\bar{\pi}$ );
          return ( $\bar{\pi}, \bar{\varphi}$ )
        end
      else  $\pi^\circ := 0, \pi^* := \bar{\pi}$ 
      end
    else
      begin
         $\pi^\circ := 0, \pi := 1$ ;
        while MaxSFP( $\pi$ ) is feasible do
          begin
             $\pi^\circ := \pi, \pi := 2\pi$ 
          end;
           $\pi^* := \pi$ 
        end;
        repeat
         $\pi := \lceil (\pi^\circ + \pi^*)/2 \rceil$ ;
        if MaxSFP( $\pi$ ) is infeasible then  $\pi^* := \pi$ 
        else  $\pi^\circ := \pi$ 
        until  $\pi^* = \pi^\circ + 1$ ;
         $\pi^* := \pi^\circ$ ;
         $\varphi^* :=$  optimal flow for MaxSFP( $\pi^*$ );
        return ( $\pi^*, \varphi^*$ )
      end
    
```

Fig. 5. Procedure CUT_CANCEL

Moreover, this inequality is tight for $X = W$, i.e., $z(W) = \rho_\pi(W)$, which is shown as follows. Let Y be the maximal tight subset of W , where a tight set means a subset X for which the inequality (18) holds with equality. Suppose that there exists $v \in W \setminus Y$, and let U be the minimal tight subset of V containing v . Since $Y \cup U$ is also tight, the maximality of Y implies that there exists $u \in U \setminus W$. For $a = (u, v)$ we can increase the value of $\varphi(a)$ without violating the constraint $\varphi(a) \leq \psi(a)$. This does not violate (18) either, since there exists no tight X with $u \notin X$ and $v \in X$. This contradicts the optimality of φ .

For any $X \subseteq V$, $z(X) = z(X \cup W) + z(X \cap W) - z(W) \leq \rho_\pi(X \cup W) + \rho_\pi(X \cap W) - \rho_\pi(W) = g(p + \pi\chi_W + \chi_{X \cup W}) + g(p + \pi\chi_W + \chi_{X \cap W}) - g(p + \pi\chi_W) - g(p + (\pi + 1)\chi_W) \leq g(p + (\pi + 1)\chi_W + \chi_X) - g(p + (\pi + 1)\chi_W) = \rho_{\pi+1}(X)$, where the first inequality is by (18) and its tightness for $X = W$ and the second is by the

discrete midpoint convexity (6). This means $y - \partial\varphi \in \arg \min f[-p - (\pi + 1)\chi_W]$, which shows that φ is a feasible flow of MaxSFP($\pi + 1$). \square

We now show an explicit upper bound on π^\bullet . The following lemma is immediate from Lemmas 6 and 7 below.

Lemma 5. *The upper bound π^\bullet used in CUT-CANCEL satisfies $\pi^\bullet \leq 2nK$.*

Lemma 6. *At the beginning of CUT-CANCEL, $p(u) - p(v) \leq (n - 1)K$ holds for any $u \in R$ and $v \in W$, and hence we have $\bar{\pi} \leq nK$.*

Proof. At the beginning of PATH-SEARCH, p is modified by POTENTIAL-UPDATE to satisfy (9). We prove that $p(u) - p(v)$ does not exceed $(n - 1)K$ for any $u \in R$ and any $v \in W$ during an execution of PATH-SEARCH. Let $p_i, \bar{\pi}_i, \pi_i^*, R_i$ and W_i denote $p, \bar{\pi}, \pi^*, R$ and W at the beginning of the i -th call of CUT-CANCEL in one execution of PATH-SEARCH. For any $u \in R_i$ and $v \in W_i$, we check how much $p(u) - p(v)$ has changed by the $(i - 1)$ -th call of CUT-CANCEL. In case of $u \in R_{i-1}$, since $v \in W_i \subseteq W_{i-1}$ and $\pi_{i-1}^* \geq 0$, we have $p_i(u) - p_i(v) = p_{i-1}(u) - (p_{i-1}(v) + \pi_{i-1}^*) \leq p_{i-1}(u) - p_{i-1}(v)$. In the other case, we must have $u \in W_{i-1}$. Since the $(i - 1)$ -th call of CUT-CANCEL increases both $p(u)$ and $p(v)$ by π_{i-1}^* , we have $p_i(u) - p_i(v) = p_{i-1}(u) - p_{i-1}(v)$. Thus, in either case, $p(u) - p(v)$ does not increase. Therefore, we have $p(u) - p(v) \leq (n - 1)K$. This implies $\bar{\pi}_i \leq K + (n - 1)K = nK$ for any i . \square

Lemma 7. *In procedure CUT-CANCEL, if $\vec{A}_\xi(\alpha)$ is empty, then MaxSFP(π) is infeasible for $\pi > nK$.*

Proof. Suppose to the contrary that MaxSFP(π) has a feasible flow φ .

We first claim that there exists no arc from R to W in E_z for $z = y - \partial\varphi$. For any pair of $u \in R$ and $v \in W$, define $z' = z - \chi_u + \chi_v$. Since $z \in \arg \min f[-p - \pi\chi_W]$ and $p(u) - p(v) \leq (n - 1)K$, we have $f(z') - f(z) = f[-p - \pi\chi_W](z') - f[-p - \pi\chi_W](z) + \pi + p(v) - p(u) > nK - (n - 1)K = K$, which implies $z' \notin \text{dom } f$ by the definition of K , and hence $(u, v) \notin E_z$.

Let $\rho : 2^V \rightarrow \mathbf{Z}$ be the submodular function associated with the M-convex set $\text{dom } f$. Since there exists no arc in E_z from R to W , we have $z(W) = \rho(W) \geq b(\Delta^+W) - c(\Delta^-W)$, where the inequality follows from Theorem 1. Since there exists no arc in $A_\xi(\alpha)$ from R to W , we also have $c(a) - \xi(a) < \alpha$ for $a \in \Delta^-W$ and $\xi(a) - b(a) < \alpha$ for $a \in \Delta^+W$. Combining these, we obtain

$$\partial\xi(W) - z(W) \leq \alpha(|\Delta^+W| + |\Delta^-W|) \leq \alpha n(n - 1). \quad (19)$$

Since $\varphi(a) \leq \psi(a)$ for $a \in \vec{D}$ and $\psi(a) = 0$ for $a \in D$ from W to R , we have

$$\partial\psi(W) \leq \partial\varphi(W). \quad (20)$$

Furthermore, it follows from $R \cap S^-(\alpha) = \emptyset$ and $W \cap S^+(\alpha) = \emptyset$ that

$$x(v) - \partial\xi(v) > -\alpha \quad (v \in R), \quad x(v) - \partial\xi(v) < \alpha \quad (v \in W). \quad (21)$$

For $S = \{v \in V \mid x(v) < \partial\xi(v)\}$, we have

$$\begin{aligned} \|x - \partial\xi\|_1 &= -2 \sum_{v \in S} (x(v) - \partial\xi(v)) \\ &= -2 \left(\sum_{v \in R \cap S} (x(v) - \partial\xi(v)) + \sum_{v \in W \cap S} (x(v) - \partial\xi(v)) \right) \\ &\leq 2(\alpha|R| + \alpha|W| + \partial\xi(W) - x(W)) \\ &= 2(\alpha|R| + \alpha|W| + \partial\xi(W) - z(W) - \partial\varphi(W) + \partial\psi(W)) \\ &\leq 2(\alpha n + \alpha n(n-1)) = 2\alpha n^2, \end{aligned}$$

where the first inequality is due to (21) and the second is to (19) and (20). This contradicts the fact that PATH_SEARCH is called only when $\|x - \partial\xi\|_1 > 2\alpha n^2$. \square

3.3 Time Complexity

We give a bound on the number of augmentations in any α -scaling phase.

Lemma 8. *The number of augmentations in any α -scaling phase is $O(n^2)$.*

Proof. When an α -scaling phase ends, we have $\|x - \partial\xi\|_1 \leq 2\alpha n^2$. Then we reduce α by a factor of two and start a new α -scaling phase. At the start of a new α -scaling phase, we modify ψ to satisfy the capacity constraints (7) for the new value of α , and modify x to maintain (8). We also modify ξ to resolve the possible violation of $(A[\alpha])$ for arcs a with residual capacity $\alpha \leq r(a) < 2\alpha$. After these modifications, we have $\|x - \partial\xi\|_1 \leq 4\alpha n^2 + 2\alpha n^2 + 2\alpha m \leq 8\alpha n^2$. Since each augmentation reduces $\|x - \partial\xi\|_1$ by 2α , the total number of augmentations in any α -scaling phase is at most $4n^2 = O(n^2)$. \square

Since the algorithm calls PATH_SEARCH before each augmentation, Lemma 8 implies the same $O(n^2)$ bound on the number of calls of PATH_SEARCH in each scaling phase. We now provide the running time bound of PATH_SEARCH. Let F denote the time for evaluating f .

Lemma 9. *Procedure PATH_SEARCH runs in $O(Fn^4 \log L \log(nK))$ time.*

Proof. At the start of PATH_SEARCH, shortest path distances can be computed in $O(n^3)$ time by the Ford-Bellman algorithm.

By Lemma 3, PATH_SEARCH calls CUT_CANCEL at most $n-1$ times. In each execution of CUT_CANCEL, we need $O(\log(nK))$ iterations for the bisection procedure since $\pi^\bullet \leq 2nK$ by Lemma 5. Each feasibility check in CUT_CANCEL requires to solve MaxSFP(π), which takes $O(n^3h)$ time by the algorithm of Fujishige and Zhang [8], where h denotes the time to compute an exchange capacity. In our framework of using a function evaluation oracle for the M-convex function, we have $h = O(F \log L)$ by the binary search method. Thus, procedure CUT_CANCEL runs in $O(Fn^3 \log L \log(nK))$ time. \square

At the start of algorithm CAPACITY_SCALING we can find a minimizer of an M-convex function in $O(Fn^3 \log L)$ time [20,21]. Since we initially set $\alpha = 2^{\lfloor \log M \rfloor}$ with $M \leq 2L$, after $O(\log L)$ scaling phases, we have $\alpha = 1$. By Lemmas 8 and 9, each scaling phase runs in $O(Fn^6 \log L \log(nK))$ time. Thus, the algorithm performs $O(\log L)$ scaling phases in $O(Fn^6(\log L)^2 \log(nK))$ time.

At the end of the 1-scaling phase, we have $\|y - \partial\xi\|_1 \leq \|x - \partial\xi\|_1 + \|\partial\psi\|_1 \leq 2n^2 + n(n-1) \leq 3n^2$. We then call SSP(ξ, y, p), which performs at most $3n^2/2$ augmentations to obtain an optimal flow ξ of MCSFP. Since each construction of the auxiliary graph can be done by $O(n^2)$ evaluations of f , the running time of this postprocess is $O(Fn^4)$, which is dominated by the above $O(Fn^6(\log L)^2 \log(nK))$ bound. Thus we obtain the following theorem.

Theorem 3. *Algorithm CAPACITY_SCALING solves the M-convex submodular flow problem MCSFP in $O(Fn^6(\log L)^2 \log(nK))$ time.*

Acknowledgements. The authors are grateful to Akihisa Tamura for helpful comments on the manuscript.

References

1. Cunningham, W.H., Frank, A.: A primal-dual algorithm for submodular flows. *Math. Oper. Res.* **10** (1985) 251–262
2. Dress, A.W.M., Wenzel, W.: Valuated matroids. *Adv. Math.* **93** (1992) 214–250
3. Edmonds, J., Giles, R.: A min-max relation for submodular functions on graphs. *Ann. Discrete Math.* **1** (1977) 185–204
4. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM* **19** (1972) 248–264
5. Fleischer, L., Iwata, S., McCormick, S.T.: A faster capacity scaling algorithm for minimum cost submodular flow. *Math. Program., Ser. A* **92** (2002) 119–139
6. Frank, A.: Finding feasible vectors of Edmonds-Giles Polyhedra. *J. Comb. Theory, Ser. B* **36** (1984) 221–239
7. Fujishige, S.: Submodular Functions and Optimization. North-Holland (1991)
8. Fujishige, S., Zhang, X.: New algorithms for the intersection problem of submodular systems. *Japan J. Indust. Appl. Math.* **9** (1992) 369–382
9. Iwata, S.: A capacity scaling algorithm for convex cost submodular flows. *Math. Program.* **76** (1997) 299–308
10. Iwata, S.: A faster scaling algorithm for minimizing submodular functions. *SIAM J. Comput.* **32** (2003) 833–840
11. Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM* **48** (2001) 761–777
12. Iwata, S., Shigeno, M.: Conjugate scaling algorithm for Fenchel-type duality in discrete convex optimization. *SIAM J. Optim.* **13** (2003) 204–211
13. Lovász, L.: Submodular functions and convexity. In: Bachem, A., Grötschel, M., Korte, B. (eds.): Mathematical Programming—The State of the Art. Springer-Verlag (1983) 235–257
14. Moriguchi, S., Murota, K.: Capacity scaling algorithm for scalable M-convex submodular flow problems. *Optim. Methods Softw.* **18** (2003) 207–218
15. Murota, K.: Valuated matroid intersection, I: optimality criteria. *SIAM J. Discrete Math.* **9** (1996) 545–561

16. Murota, K.: Submodular flow problem with a nonseparable cost function. *Combinatorica* **19** (1999) 87–109
17. Murota, K.: Discrete Convex Analysis. SIAM (2003)
18. Murota, K., Tamura, A.: Application of M-convex submodular flow problem to mathematical economics. *Japan J. Indust. Appl. Math.* **20** (2003) 257–277
19. Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory, Ser. B* **80** (2000) 346–355
20. Shioura, A.: Fast scaling algorithms for M-convex function minimization with application to resource allocation problem. *Discrete Appl. Math.* **134** (2003) 303–316
21. Tamura, A.: Coordinatewise domain scaling algorithm for M-convex function minimization. In: Cook, W.J., Schulz, A.S. (eds.): Integer Programming and Combinatorial Optimization. LNCS **2337**. Springer-Verlag (2002) 21–35

Integer Concave Cocirculations and Honeycombs

Alexander V. Karzanov

Institute for System Analysis, 9, Prospect 60 Let Oktyabrya, 117312 Moscow, Russia
sasha@cs.isa.ac.ru

Abstract. A *convex triangular grid* is a planar digraph G embedded in the plane so that each bounded face is an equilateral triangle with three edges and their union \mathcal{R} forms a convex polygon. A function $h : E(G) \rightarrow \mathbb{R}$ is called a *concave cocirculation* if $h(e) = g(v) - g(u)$ for each edge $e = (u, v)$, where g is a concave function on \mathcal{R} which is affinely linear within each bounded face of G . Knutson and Tao obtained an integrality result on so-called *honeycombs* implying that if an integer-valued function on the boundary edges is extendable to a concave cocirculation, then it is extendable to an integer one.

We show a sharper property: for any concave cocirculation h , there exists an integer concave cocirculation h' satisfying $h'(e) = h(e)$ for each edge e with $h(e) \in \mathbb{Z}$ contained in the boundary or in a bounded face where h is integer on all edges.

Also relevant polyhedral and algorithmic results are presented.

Keywords: Planar graph, Discrete convex (concave) function, Honeycomb

AMS Subject Classification: 90C10, 90C27, 05C99

1 Introduction

Knutson and Tao [4] proved a conjecture concerning highest weight representations of $GL_n(\mathbb{C})$. They used one combinatorial model, so-called *honeycombs*, and an essential part of the whole proof was to show the existence of an integer honeycomb under prescribed integer boundary data. The obtained integrality result on honeycombs admits a re-formulation in terms of discrete concave functions on triangular grids in the plane.

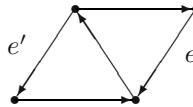
The main purpose of this paper is to show a sharper integrality property for discrete concave functions.

We start with basic definitions. Let ξ_1, ξ_2, ξ_3 be three affinely independent vectors in the euclidean plane \mathbb{R}^2 , whose sum is the zero vector. By a *convex (triangular) grid* we mean a finite planar digraph $G = (V(G), E(G))$ embedded in the plane such that: (a) each bounded face of G is a triangle surrounded by three edges and each edge (u, v) satisfies $v - u \in \{\xi_1, \xi_2, \xi_3\}$, and (b) the region $\mathcal{R} = \mathcal{R}(G)$ of the plane spanned by G is a convex polygon. In this paper a convex grid can be considered up to an affine transformation, and to visualize objects and constructions in what follows, we will fix the generating vectors ξ_1, ξ_2, ξ_3 as $(1, 0)$, $(-1, \sqrt{3})/2$, $(-1, -\sqrt{3})/2$, respectively. Then each bounded face is an

equilateral triangle (a *little triangle* of G) surrounded by a directed circuit with three edges (a *3-circuit*). When \mathcal{R} forms a (big) triangle, we call G a *3-side grid* (this case is most popular in applications).

A real-valued function h on the edges of G is called a *cocirculation* if the equality $h(e) + h(e') + h(e'') = 0$ holds for each 3-circuit formed by edges e, e', e'' . This is equivalent to the existence of a function g on \mathcal{R} which is affinely linear within each bounded face and satisfies $h(e) = g(v) - g(u)$ for each edge $e = (u, v)$. Such a g is determined up to adding a constant, and we refer to h as a *concave cocirculation* if g is concave. (The restriction of such a g to $V(G)$ is usually called a *discrete concave* function.) It is easy to see that a cocirculation h is concave if and only if each *little rhombus* ρ (the union of two little triangles sharing a common edge) satisfies the following *rhombus condition*:

- (RC) $h(e) \geq h(e')$, where e, e' are non-adjacent (parallel) edges in ρ , and e enters an obtuse vertex of ρ .



Let $B(G)$ denote the set of edges in the boundary of G . Concave cocirculations are closely related (via Fenchel's type transformations) to honeycombs, and Knutson and Tao's integrality result on the latter is equivalent to the following.

Theorem 1. [4] *For a convex grid G and a function $h_0 : B(G) \rightarrow \mathbb{Z}$, there exists an integer concave cocirculation in G coinciding with h_0 on $B(G)$ if h_0 is extendable to a concave cocirculation in G at all.*

For a direct proof of this theorem for 3-side grids, without appealing to honeycombs, see Buch [1]. (Note also that for a 3-side grid G , a combinatorial characterization for the set of functions h_0 on $B(G)$ extendable to concave cocirculations is given in [5]: it is a polyhedral cone in $\mathbb{R}^{B(G)}$ whose nontrivial facets are described by Horn's type inequalities with respect to so-called *puzzles*; for an alternative proof and an extension to arbitrary convex grids, see [3].)

In this paper we extend Theorem 1 as follows.

Theorem 2. *Let h be a concave cocirculation in a convex grid G . There exists an integer concave cocirculation h' in G such that $h'(e) = h(e)$ for all edges $e \in O_h \cup I_h$. Here O_h is the set of boundary edges e where $h(e)$ is an integer, and I_h is the set of edges contained in little triangles Δ such that h takes integer values on the three edges of Δ .*

Remark 1. One could attempt to further strengthen Theorem 1 by asking: can one improve any concave cocirculation h to an integer one preserving the values on *all* edges where h is already integral? In general, the answer is negative; a counterexample will be given in the end of this paper.

Our method of proof of Theorem 2 is constructive and based on iteratively transforming the current concave cocirculation until the desired integer concave cocirculation is found. As a consequence, we obtain a polynomial-time combinatorial algorithm to improve h to h' as required. (The idea of proof of Theorem 1 in [1] is to show the existence of a concave cocirculation coinciding with h_0 on $B(G)$ whose values are expressed as integer combinations of values of h_0 ; [4] establishes an analogous property for honeycombs. Our approach is different.) We prefer to describe an iteration by considering a corresponding task on the related honeycomb model and then translating the output to the language of cocirculations in G , as this makes our description technically simpler and more enlightening.

The above theorems admit a re-formulation in polyhedral terms. Given a subset $F \subseteq E(G)$ and a function $h_0 : F \rightarrow \mathbb{R}$, let $\mathcal{C}(G, h_0)$ denote the set of concave cocirculations in G such that $h(e) = h_0(e)$ for all $e \in F$. Since concave cocirculations are described by linear constraints, $\mathcal{C}(G, h_0)$ forms a (possibly empty) polyhedron in $\mathbb{R}^{E(G)}$. Then Theorem 1 says that such a polyhedron (if nonempty) has an integer point h in the case $h_0 : B(G) \rightarrow \mathbb{Z}$, whereas Theorem 2 is equivalent to saying that a similar property takes place if $h_0 : F \rightarrow \mathbb{Z}$ and $F = B' \cup \cup(E(\Delta) : \Delta \in T)$, where $B' \subseteq B(G)$ and T is a set of little triangles. (Note that when $F = B(G)$ and when $\mathcal{R}(G)$ is not a hexagon, one can conclude from the concavity that $\mathcal{C}(G, h_0)$ is bounded, i.e., it is a polytope.)

On the “negative” side, it turned out that $\mathcal{C}(G, h_0)$ with $h_0 : B(G) \rightarrow \mathbb{Z}$ need not be an integer polytope; an example with a half-integer but not integer vertex is given in [4] (and in [1]). One can show that the class of such polytopes has “unbounded fractionality”. Moreover, denominators of vertex entries can be arbitrarily increasing as the size of G grows even if functions h_0 with smaller domains are considered. Hereinafter by the *size* of G we mean its maximum side length (=number of edges). We show the following.

Theorem 3. *For any positive integer k , there exists a 3-side grid G of size $O(k)$ and a function $h_0 : F \rightarrow \mathbb{Z}$, where F is the set of edges on two sides of G , such that $\mathcal{C}(G, h_0)$ has a vertex h satisfying: (a) $h(e)$ has denominator k for some edge $e \in E(G)$, and (b) h takes integer values on the third side.*

(In this case $\mathcal{C}(G, h_0)$ is also a polytope, and if h'_0 is the restriction of h to a set $F' \supset F$, then h is a vertex of $\mathcal{C}(G, h'_0)$ as well.)

This paper is organized as follows. In Section 2 we explain the notion of honeycomb and a relationship between honeycombs and concave cocirculations. Sections 3 and 4 consider special paths and cycles P in a honeycomb and describe a certain transformation of the honeycomb in a neighbourhood of P . Section 5 uses such transformations to “improve” the honeycomb and eventually proves Theorem 2. A construction of G, h_0 proving Theorem 3 is given in Section 6; it relies on an approach involving honeycombs as well. We also explain there (in Remark 3) that the set F in this theorem can be reduced further. The concluding Section 7 outlines algorithmic aspects, suggests a slight strengthening of Theorem 2 and gives a counterexample mentioned in Remark 1.

2 Honeycombs

For technical needs of this paper, our definition of honeycombs will be somewhat different from, though equivalent to, that given in [4]. It is based on a notion of pre-honeycombs, and before introducing the latter, we clarify some terminology and notation user later on. Let ξ_1, ξ_2, ξ_3 be the generating vectors as above (note that they follow anticlockwise around the origin).

The term *line* is applied to (*fully*) *infinite*, *semiinfinite*, and *finite* lines, i.e., to sets of the form $a + \mathbb{R}b$, $a + \mathbb{R}_+b$, and $\{a + \lambda b : 0 \leq \lambda \leq 1\}$, respectively, where $a \in \mathbb{R}^2$ and $b \in \mathbb{R}^2 \setminus \{\mathbf{0}\}$. For a vector (point) $v \in \mathbb{R}^2$ and $i = 1, 2, 3$, we denote by $\Xi_i(v)$ the infinite line containing v and perpendicular to ξ_i , i.e., the set of points u with $(u - v) \cdot \xi_i = 0$. (Hereinafter $x \cdot y$ denotes the inner product of vectors x, y .) The line $\Xi_i(v)$ is the union of two semiinfinite lines $\Xi_i^+(v)$ and $\Xi_i^-(v)$ with the end v , where the rays $\Xi_i^+(v) - v, \mathbb{R}_+\xi_i$ and $\Xi_i^-(v) - v$ follow in the *anticlockwise* order around the origin. Any line perpendicular to ξ_i is called a Ξ_i -*line*.

By a Ξ -*system* we mean a finite set \mathcal{L} of Ξ_i -lines ($i \in \{1, 2, 3\}$) along with an *integer* weighting w on them. For a point $v \in \mathbb{R}^2$, a “sign” $s \in \{+, -\}$, and $i = 1, 2, 3$, define $w_i^s(v)$ to be the sum of weights $w(L)$ of the lines $L \in \mathcal{L}$ whose intersection with $\Xi_i^s(v)$ contains v and is a line (not a point). We call a Ξ -system (\mathcal{L}, w) a *pre-honeycomb* if for any point v , the numbers $w_i^s(v)$ are *nonnegative* and satisfy the condition

$$w_1^+(v) - w_1^-(v) = w_2^+(v) - w_2^-(v) = w_3^+(v) - w_3^-(v) =: \text{div}_w(v); \quad (1)$$

$\text{div}_w(v)$ is called the *divergency* at v .

Now a *honeycomb* is a certain non-standard edge-weighted planar graph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$ with vertex set $\mathcal{V} \neq \emptyset$ and edge set \mathcal{E} in which non-finite edges are allowed. More precisely: (i) each vertex is incident with at least 3 edges; (ii) each edge is a line with no interior point contained in another edge; (iii) $w(e)$ is a *positive* integer for each $e \in \mathcal{E}$; and (iv) (\mathcal{E}, w) is a pre-honeycomb. Then each vertex v has degree at most 6, and for $i = 1, 2, 3$ and $s \in \{+, -\}$, we denote by $e_i^s(v)$ the edge incident to v and contained in $\Xi_i^s(v)$ when such an edge exists, and say that this edge *has sign* s at v . In [4] the condition on a vertex v of a honeycomb similar to (1) is called the *zero-tension condition*, motivated by the observation that if each edge incident to v pulls on v with a tension equal to its weight, then the total force applied to v is zero. Figure 1 illustrates a honeycomb with three vertices u, v, z and ten edges of which seven are semiinfinite.

We will take advantage of the fact that any pre-honeycomb (\mathcal{L}, w) determines, in a natural way, a unique honeycomb $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w')$ with $w_i^s(v) = (w')_i^s(v)$ for all v, s, i . Here \mathcal{V} is the set of points v for which at least three numbers among $w_i^s(v)$'s are nonzero. The set \mathcal{E} consists of all maximal Ξ_i -lines e for $i = 1, 2, 3$ such that any interior point v on e satisfies $w_i^+(v) > 0$ and does not belong to \mathcal{V} ; the weight $w'(e)$ is defined to be just this number $w_i^+(v)$ ($= w_i^-(v)$), which does not depend on v .

Since $\mathcal{V} \neq \emptyset$, one can conclude from (1) that a honeycomb has no fully infinite edge but the set of semiinfinite edges in it is nonempty. This set, called

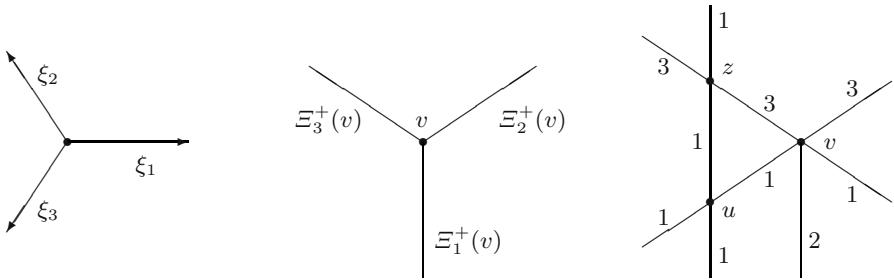


Fig. 1. Generating vectors, lines $\Xi_i^+(v)$, and a honeycomb instance

the *boundary* of \mathcal{H} and denoted by $\mathcal{B}(\mathcal{H})$, is naturally partitioned into subsets \mathcal{B}_i^s consisting of the semiinfinite edges of the form $\Xi_i^s(\cdot)$. Then (1) implies that $w(\mathcal{B}_i^+) - w(\mathcal{B}_i^-)$ is the same for $i = 1, 2, 3$. (For a subset $E' \subseteq E$ and a function $c : E \rightarrow \mathbb{R}$, $c(E')$ stands for $\sum(c(e) : e \in E')$.)

Let us introduce the *dual coordinates* d_1, d_2, d_3 of a point $x \in \mathbb{R}^2$ by setting

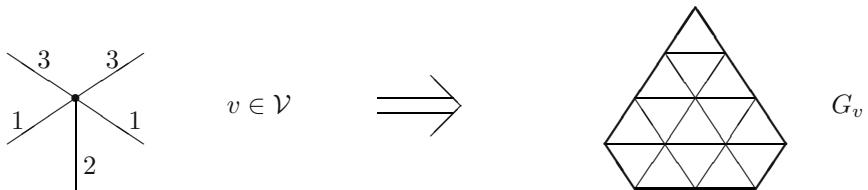
$$d_i(x) := -x \cdot \xi_i, \quad i = 1, 2, 3.$$

Since $\xi_1 + \xi_2 + \xi_3 = 0$, one has

$$d_1(x) + d_2(x) + d_3(x) = 0 \quad \text{for each } x \in \mathbb{R}^2. \quad (2)$$

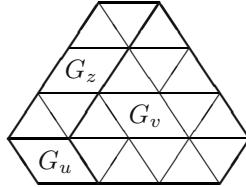
When one traverses an edge e of \mathcal{H} , one dual coordinate remains constant while the other two trade off; this *constant dual coordinate* is denoted by $d^c(e)$.

Next we explain that the honeycombs $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$ one-to-one correspond to the concave cocirculations via a sort of planar duality. Let $v \in \mathcal{V}$. Since the numbers $w_i^s(v)$ are nonnegative, condition (1) is equivalent to the existence of a (unique) grid G_v whose boundary is formed, in the anticlockwise order, by $w_1^+(v)$ edges parallel to ξ_1 , followed by $w_3^-(v)$ edges parallel to ξ_3 , followed by $w_2^+(v)$ edges parallel to ξ_2 , and so on, as illustrated in the picture.



For an edge e incident to v , label e^* the corresponding side of G_v . For each finite edge $e = uv$ of \mathcal{H} , glue together the grids G_u and G_v by identifying the sides labelled e^* in both. One can see that the resulting graph G is a convex grid. Also each nonempty set \mathcal{B}_i^s one-to-one corresponds to a side of G , denoted by B_i^s ; it is formed by $w(\mathcal{B}_i^s)$ edges parallel to ξ_i , and the outward normal at B_i^s

points at the direction (i, s) . The next picture illustrates the grid generated by the honeycomb in Fig. 1.



The dual coordinates of vertices of \mathcal{H} generate function h on $E(G)$ defined by:

$$h(e) := d_i(v), \quad e \text{ an edge in } G_v \text{ parallel to } \xi_i, v \in \mathcal{V}, i = 1, 2, 3. \quad (3)$$

Then (2) implies $h(e) + h(e') + h(e'') = 0$ for any little triangle with edges e, e', e'' in G , i.e., h is a cocirculation. To see that h is concave, it suffices to check (RC) for a little rhombus ρ formed by little triangles lying in different graphs G_u and G_v . Then $\tilde{e} = uv$ is an edge of \mathcal{H} ; let \tilde{e} be perpendicular to ξ_i and assume $\tilde{e} = e_i^-(u) = e_i^+(v)$. Observe that $d_{i+1}(u) > d_{i+1}(v)$ (taking indices modulo 3) and that the side-edge e of ρ lying in G_u and parallel to ξ_{i+1} enters an obtuse vertex of ρ . Therefore, $h(e) = d_{i+1}(u) > d_{i+1}(v) = h(e')$, where e' is the side-edge of ρ parallel to e (lying in $G(v)$), as required.

Conversely, let h be a concave cocirculation in a convex grid G . Subdivide G into maximal subgraphs G_1, \dots, G_k , each being the union of little triangles where, for each $i = 1, 2, 3$, all edges parallel to ξ_i have the same value of h . The concavity of h implies that each G_j is again a convex grid; it spans a maximal region where the corresponding function g on \mathcal{R} is affinely linear, called a *flatspace* of h . For $j = 1, \dots, k$, take point v_j with the dual coordinates $d_i(v_j) = h(e_i)$, $i = 1, 2, 3$, where $e_i \in E(G_j)$ is parallel to ξ_i . (Then $h(e_1) + h(e_2) + h(e_3) = 0$ implies (2), so v_j exists; also the points v_j are different). For each pair of graphs $G_j, G_{j'}$ having a common side S , connect v_j and $v_{j'}$ by line (finite edge) ℓ ; observe that ℓ is perpendicular to S . And if a graph G_j has a side S contained in the boundary of G , assign the semiinfinite edge $\ell = \Xi_i^s(v_j)$ whose direction (i, s) corresponds to the outward normal at S . In all cases assign the weight of ℓ to be the number of edges in S . One can check (by reversing the argument above) that the obtained sets of points and weighted lines constitute a honeycomb \mathcal{H} , and that the above construction for \mathcal{H} returns G, h .

3 Legal Paths and Cycles

In this section we consider certain paths (possibly cycles) in a honeycomb $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$. A transformation of \mathcal{H} with respect to such a path, described in the next section, “improves” the honeycomb, in a certain sense, and we will show that a series of such improvements results in a honeycomb determining an integer concave cocirculation as required in Theorem 2. First of all we need some definitions and notation.

Let \mathcal{V}^\bullet denote the set of vertices $v \in \mathcal{V}$ having at least one nonintegral dual coordinate $d_i(v)$, and \mathcal{E}^\bullet the set of edges $e \in \mathcal{E}$ whose constant coordinate $d^c(e)$ is nonintegral. For brevity we call such vertices and edges *nonintegral*.

For $s \in \{+, -\}$, $-s$ denotes the sign opposite to s . Let $v \in \mathcal{V}$. An edge $e_i^s(v)$ is called *dominating* at v if $w_i^s(v) > w_i^{-s}(v)$. By (1), v has either none or three dominating edges, each pair forming an angle of 120° . A pair $\{e_i^s(v), e_j^{s'}(v)\}$ of distinct *nonintegral* edges is called *legal* if either they are *opposite* to each other at v , i.e., $j = i$ and $s' = -s$, or both edges are dominating at v (then $j \neq i$ and $s' = s$). By a *path* in \mathcal{H} we mean a finite alternating sequence $P = (v_0, q_1, v_1, \dots, q_k, v_k)$, $k \geq 1$, of vertices and edges where: for $i = 2, \dots, k-1$, q_i is a finite edge and v_{i-1}, v_i are its ends; q_1 is either a finite edge with the ends v_0, v_1 , or a semiinfinite edge with the end v_1 ; similarly, if $k > 1$ then q_k is either a finite edge with the ends v_{k-1}, v_k , or a semiinfinite edge with the end v_{k-1} . When q_1 is semiinfinite, v_0 is thought of as a *dummy* (“infinite”) vertex, and we write $v_0 = \{\emptyset\}$; similarly, $v_k = \{\emptyset\}$ when q_k is semiinfinite and $k > 1$. Self-intersecting paths are admitted. We call P

- (i) an *open path* if $k > 1$ and both edges q_1, q_k are semiinfinite;
- (ii) a *legal path* if each pair of consecutive edges q_i, q_{i+1} is legal;
- (iii) a *legal cycle* if it is a legal path, $v_0 = v_k \neq \{\emptyset\}$ and $\{q_k, q_1\}$ is a legal pair.

A legal cycle P is considered up to shifting cyclically and the indices are taken modulo k . We say that a legal path P *turns* at $v \in \mathcal{V}$ if, for some i with $v_i = v$, the (existing) edges q_i, q_{i+1} are not opposite at v_i (then q_i, q_{i+1} are dominating at v_i). We also call such a triple (q_i, v_i, q_{i+1}) a *bend* of P at v .

Assume \mathcal{V}^\bullet is nonempty. (When $\mathcal{V}^\bullet = \emptyset$, the concave cocirculation in G determined by \mathcal{H} is already integral.) A trivial but important observation from (2) is that if a vertex v has a nonintegral dual coordinate, then it has at least two such coordinates. This implies $\mathcal{E}^\bullet \neq \emptyset$. Moreover, if $e \in \mathcal{E}^\bullet$ is dominating at v , then e forms a legal pair with another nonintegral edge dominating at v .

Our method of proof of Theorem 2 will rely on the existence of a legal path with some additional properties, as follows.

Lemma 1. *There exists an open legal path or a legal cycle $P = (v_0, q_1, v_1, \dots, q_k, v_k)$ such that:*

- (i) *each edge e of \mathcal{H} occurs in P at most twice, and if it occurs exactly twice, then P traverses e in both directions, i.e., $e = q_i = q_j$ and $i < j$ imply $v_i = v_{j-1}$;*
- (ii) *if an edge e occurs in P twice, then $w(e) > 1$;*
- (iii) *for each vertex v of \mathcal{H} , the number of times P turns at v does not exceed $\min\{2, |\text{div}_w(v)|\}$.*

Proof. We grow a legal path P , step by step, by the following process. Initially, choose a nonintegral semiinfinite edge e if it exists, and set $P = (v_0, q_1, v_1)$, where $v_0 := \{\emptyset\}$, $q_1 := e$, and v_1 is the end of e . Otherwise we start with $P = (v_0, q_1, v_1)$, where q_1 is an arbitrary nonintegral finite edge and v_0, v_1 are its ends. Let $P = (v_0, q_1, v_1, \dots, q_i, v_i)$ be a current legal path with $v := v_i \in \mathcal{V}$ satisfying (i),(ii),(iii). At an iteration, we wish either to increase P by adding some q_{i+1}, v_{i+1} (maintaining (i),(ii),(iii)) or to form the desired cycle.

By the above observation, $e := q_i$ forms a legal pair with at least one edge e' incident to v . We select such an e' by rules specified later and act as follows. Suppose e' occurs in P and is traversed from v , i.e., $v = v_{j-1}$ and $e' = e_j$ for some $j < i$. Then the part of P from v_{j-1} to v_i forms a legal cycle; we finish the process and output this cycle. Clearly it satisfies (i) and (ii) (but the number of bends at v may increase). Now suppose e' is not traversed from v . Then we grow P by adding e' as the new last edge q_{i+1} and adding v_{i+1} to be the end of e' different from v if e' is finite, and to be $\{\emptyset\}$ if e' is semiinfinite. In the latter case, the new P is an open legal path (taking into account the choice of the first edge q_1); we finish and output this P . And in the former case, we continue the process with the new current P . Clearly property (i) is maintained.

We have to show that e' can be chosen so as to maintain the remaining properties (concerning e' and v). Consider two cases.

Case 1. e is not dominating at v . Then e' is opposite to e at v (as the choice is unique), and (iii) remains valid as no new bend at v arises. If e' is dominating at v , then $w(e') > w(e) \geq 1$, implying (ii). And if e' is not dominating at v , then $w(e') = w(e)$ and, obviously, the new P traverses e' as many times as it traverses e , implying (ii) as well.

Case 2. e is dominating at v . Let the old P turn b times at v . First suppose P has a bend $\beta = (q_j, v_j, q_{j+1})$ at v not using the edge e . Since the edges occurring in any bend are nonintegral and dominating at the corresponding vertex, $\{e, q_{j+1}\}$ is a legal pair. We choose e' to be q_{j+1} . This leads to forming a cycle with b bends at v as before (as the bend β is destroyed while the only bend (e, v, e') is added), implying (iii).

So assume such a β does not exist. Then the old P can have at most one bend at v , namely, one of the form $\beta' = (q, v, e)$, whence $b \leq 1$. If $b < |\text{div}_w(v)|$, then taking as e' a nonintegral edge dominating at v and different from e maintains both (ii) and (iii) (to see (ii), observe that the number of times P traverses e' is less than $w(e')$). Now let $b = |\text{div}_w(v)|$. Then $b = 1$ (as $\text{div}_w(v) \neq 0$) and P has the bend β' as above. Therefore, P traverses e twice (in β' and as q_i), and we conclude from this fact together with $|\text{div}_w(v)| = 1$ that e has the opposite edge \bar{e} at v . Moreover, \bar{e} cannot occur in P . For otherwise P would traverse e more than twice, taking into account that \bar{e} forms a legal pair only with e (as \bar{e} is non-dominating at v). Thus, the choice of e' to be \bar{e} maintains (ii) and (iii), completing the proof of the lemma. \square

4 ϵ -Deformation

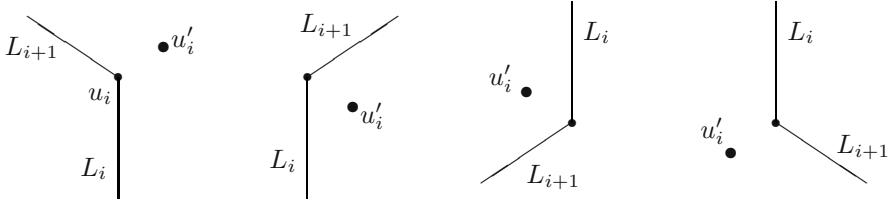
Let $P = (v_0, q_1, v_1, \dots, q_k, v_k)$ be as in Lemma 1. Our transformation of the honeycomb $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$ in question is, roughly speaking, a result of “moving a unit weight copy of P in a normal direction” (considering P as a curve in the plane); this is analogous to an operation on more elementary paths or cycles of honeycombs in [4,5]. It is technically easier for us to describe such a transformation by handling a pre-honeycomb behind \mathcal{H} in which the line set include the maximal straight subpaths of P .

When (q_i, v_i, q_{i+1}) is a bend, we say that v_i is a *bend vertex* of P . We assume that v_0 is a bend vertex if P is a cycle. For a bend vertex v_i , we will distinguish between the cases when P turns right and turns left at v_i , defined in a natural way regarding the orientation of P . Let $v_{t(0)}, v_{t(1)}, \dots, v_{t(r)}$ ($0 = t(0) < t(1) < \dots < t(r) = k$) be the sequence of bend or dummy vertices of P . Then for $i = 1, \dots, r$, the union of edges $q_{t(i-1)+1}, \dots, q_{t(i)}$ is a (finite, semiinfinite or even fully infinite) line, denoted by L_i . For brevity $v_{t(i)}$ is denoted by u_i .

Our transformation of \mathcal{H} depends on a real parameter $\epsilon > 0$ measuring the distance of moving P and on a direction of moving; let for definiteness we wish to move P “to the right” (moving “to the left” is symmetric). We assume that ϵ is small enough; an upper bound on ϵ will be discussed later. By the transformation, a unit weight copy of each line L_i (considered as oriented from u_{i-1} to u_i) is split off the honeycomb and moves (possibly extending or shrinking) at distance ϵ to the right, turning into a parallel line L'_i connecting u'_{i-1} and u'_i . Let us describe this construction more formally. First, for a bend vertex u_i , let the constant coordinates of the lines L_i and L_{i+1} be p -th and p' -th dual coordinates, respectively. Then the point u'_i is defined by

$$\begin{aligned} d_p(u'_i) &:= d_p(u_i) - \epsilon, & d_{p'}(u'_i) &:= d_{p'}(u_i) + \epsilon & \text{if } L_i, L_{i+1} \text{ have sign } + \text{ at } v, \\ d_p(u'_i) &:= d_p(u_i) + \epsilon, & d_{p'}(u'_i) &:= d_{p'}(u_i) - \epsilon & \text{if } L_i, L_{i+1} \text{ have sign } - \text{ at } v, \end{aligned} \quad (4)$$

where, similar to edges, a Ξ_i -line is said to have sign s at its end v if it is contained in $\Xi_i^s(v)$. Some possible cases are illustrated in the picture.



Second, for $i = 1, \dots, r$, define L'_i to be the line connecting u'_{i-1} and u'_i (when P is an open path, u'_0 and u'_r are dummy points and the non-finite lines L'_1, L'_r are defined in a natural way). Denoting by $\ell(L)$ the Euclidean length (scaled by $2/\sqrt{3}$) of a line L , one can see that if a line L_i is finite and $\epsilon \leq \ell(L_i)$, then

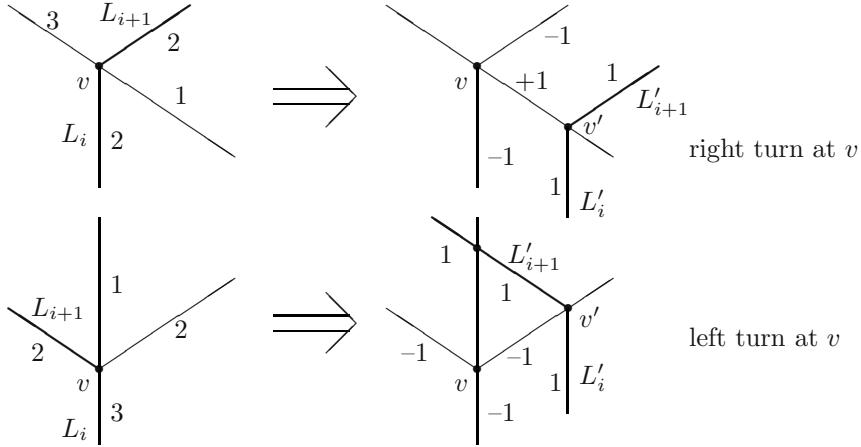
$$\begin{aligned} \ell(L'_i) &= \ell(L_i) - \epsilon & \text{if } P \text{ turns right at both } u_{i-1} \text{ and } u_i, \\ \ell(L'_i) &\geq \ell(L_i) & \text{otherwise.} \end{aligned} \quad (5)$$

This motivates a reasonable upper bound on ϵ , to be the minimum length $\bar{\epsilon}_0$ of an L_i such that P turns right at both u_{i-1}, u_i ($\bar{\epsilon}_0 = \infty$ when no such L_i exists).

Third, consider the Ξ -system $\mathcal{P} = (\{L_1, \dots, L_r\} \cup \mathcal{E}, \tilde{w})$, where $\tilde{w}(L_i) = 1$ for $i = 1, \dots, r$, and $\tilde{w}(e)$ is equal to $w(e)$ minus the number of occurrences of $e \in \mathcal{E}$ in L_1, \dots, L_r . This \mathcal{P} is a pre-honeycomb representing \mathcal{H} , i.e., satisfying $w_i^s(v) = \tilde{w}_i^s(v)$ for all v, i, s . We replace in \mathcal{P} the lines L_1, \dots, L_r by the lines L'_1, \dots, L'_r with unit weight each. (When $\epsilon = \bar{\epsilon}_0 < \infty$, the length of at least one

line L'_i reduces to zero, by (5), and this line vanishes in \mathcal{P}' .) Also for each bend vertex u_i , we add line R_i connecting u_i and u'_i . We assign to R_i weight 1 if P turns right at u_i , and -1 otherwise. Let $\mathcal{P}' = (\mathcal{L}', w')$ be the resulting Ξ -system.

The transformation in a neighbourhood of a bend vertex $v = u_i$ is illustrated in the picture; here the numbers on edges indicate their original weights or the changes due to the transformation, and (for simplicity) P passes v only once.



Lemma 2. *There exists $\bar{\epsilon}_1$, $0 < \bar{\epsilon}_1 \leq \bar{\epsilon}_0$ such that \mathcal{P}' is a pre-honeycomb for any nonnegative real $\epsilon \leq \bar{\epsilon}_1$.*

Proof. Let $0 < \epsilon < \bar{\epsilon}_0$. To see that \mathcal{P}' has zero tension everywhere, it suffices to check this property at the bend vertices u_i of P and their “copies” u'_i . For a bend vertex u_i , let \mathcal{P}_i be the Ξ -system formed by the lines L_i, L_{i+1}, R_i with weights $-1, -1, w'(R_i)$, respectively, and \mathcal{P}'_i the Ξ -system formed by the lines L'_i, L'_{i+1}, R_i with weights $1, 1, w'(R_i)$, respectively. One can see that \mathcal{P}_i has zero tension at u_i and \mathcal{P}'_i has zero tension at u'_i , wherever (right or left) P turns at u_i . This implies the zero tension property for \mathcal{P}' , taking into account that \mathcal{P} has this property (as \mathcal{P}_i and \mathcal{P}'_i describe the corresponding local changes at u_i, u'_i when \mathcal{P} turns into \mathcal{P}').

It remains to explain that the numbers $(w')^s_j(v)$ are nonnegative for all corresponding v, j, s when $\epsilon > 0$ is sufficiently small.

By (i),(ii) in Lemma 1, $\tilde{w} \geq 0$ for all $e \in \mathcal{E}$. So the only case when $(w')^s_p(v)$ might be negative is when v lies on a line R_i with weight -1 and R_i is a Ξ_p -line. Note that if $u_j = u_{j'}$ for some $j \neq j'$, i.e., P turns at the corresponding vertex v of \mathcal{H} twice, then the points u'_j and $u'_{j'}$ move along different rays out of v (this can be seen from (4), taking into account (i) in Lemma 1). Hence we can choose $\epsilon > 0$ such that the interiors of the lines R_i are pairwise disjoint.

Consider R_i with $w'(R_i) = -1$, and let e be the edge dominating at the vertex $v = u_i$ and different from $q_{t(i)}$ and $q_{t(i)+1}$. Observe that the point u'_i moves just along e , so the line R_i is entirely contained in e when ϵ does not exceed the length of e . We show that $\tilde{w}(e) > 0$, whence the result follows. This

is so if the number α of lines among L_1, \dots, L_r that contain e (equal to 0, 1 or 2) is strictly less than $w(e)$. For a contradiction, suppose $\alpha = w(e)$ (as $\alpha \leq w(e)$, by Lemma 1). This implies that the number of bends of P at v using the edge e is at least $|\text{div}_w(v)|$. Then the total number of bends at v is greater than $|\text{div}_w(v)|$ (as the bend $(q_{t(i)}, u_i, q_{t(i)+1})$ does not use e), contradicting (iii) in Lemma 1. So $\alpha < w(e)$, as required. \square

For ϵ as in the lemma, \mathcal{P}' determines a honeycomb \mathcal{H}' , as explained in Section 2. We say that \mathcal{H}' is obtained from \mathcal{H} by the *right ϵ -deformation* of P .

5 Proof of Theorem 2

In what follows, vertices u, v of a honeycomb with edge weights w'' are said to have *the same sign* if $\text{div}_{w''}(u)\text{div}_{w''}(v) \geq 0$, and $|\text{div}_{w''}(v)|$ is called the *excess* at v .

Consider a concave cocirculation h in a convex grid G and the honeycomb $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$ determined by h . Define $\beta = \beta_{\mathcal{H}}$ to be the total weight of *nonintegral* semiinfinite edges of \mathcal{H} , $\delta = \delta_{\mathcal{H}}$ to be the total excess of *nonintegral* vertices of \mathcal{H} , and $\omega = \omega_{\mathcal{H}}$ to be the total weight of edges incident to *integral* vertices. We prove Theorem 2 by induction on

$$\eta := \eta_{\mathcal{H}} := \beta + \delta - \omega,$$

considering all concave cocirculations h in the given G . Observe that ω does not exceed $|E(G)|$; hence η is bounded from below. Note also that in the case $\beta = \delta = 0$, all edges of \mathcal{H} are integral (then h is integral as well). Indeed, suppose the set \mathcal{E}^\bullet of nonintegral edges of \mathcal{H} is nonempty, and let $e \in \mathcal{E}^\bullet$. Take the maximal line L that contains e and is covered by edges of \mathcal{H} . Since $d^c(L) = d^c(e)$ is not an integer and $\beta = 0$, L contains no semiinfinite edge; so L is finite. The maximality of L implies that each end v of L is a vertex of \mathcal{H} and, moreover, $\text{div}_w(v) \neq 0$. Also $v \in \mathcal{V}^\bullet$. Then $\delta \neq 0$; a contradiction.

Thus, we may assume that $\beta + \delta > 0$ and that the theorem is valid for all concave cocirculations on G whose corresponding honeycombs \mathcal{H}' satisfy $\eta_{\mathcal{H}'} < \eta_{\mathcal{H}}$. We use notation, constructions and facts from Sections 3, 4.

Choose $P = (v_0, q_1, v_1, \dots, q_k, v_k)$ as in Lemma 1. Note that if P is a cycle, then the fact that all bends of P are of the same degree 120° implies that there are two consecutive bend vertices u_i, u_{i+1} where the direction of turn of P is the same. We are going to apply to P the right ϵ -deformation, assuming that either P is an open path, or P is a cycle having two consecutive bend vertices where it turns right (for P can be considered up to reversing).

We gradually grow the parameter ϵ from zero, obtaining the (parameteric) honeycomb $\mathcal{H}' = (\mathcal{V}', \mathcal{E}', w')$ as described in Section 4. Let $\bar{\epsilon}_1$ be specified as the *maximum* real or $+\infty$, with $\bar{\epsilon}_1 \leq \bar{\epsilon}_0$, satisfying the claim in Lemma 2. (Such an $\bar{\epsilon}_1$ exists, as if $\epsilon' > 0$ and if \mathcal{P}' is a pre-honeycomb for any $0 < \epsilon < \epsilon'$, then \mathcal{P}' is a pre-honeycomb for $\epsilon = \epsilon'$ either, by continuity and compactness.) We stop

growing ϵ as soon as it reaches the bound $\bar{\epsilon}_1$ or at least one of the following events happens:

(E1) when P is an open path, the constant coordinate of some of the (semi-infinite or infinite) lines L'_1 and L'_r becomes an integer;

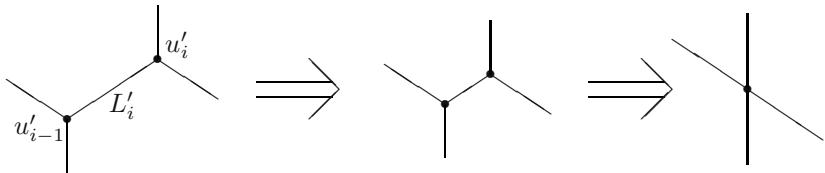
(E2) two vertices of \mathcal{H}' with different signs meet (merge);

(E3) some of the lines L'_i meets an integer vertex v of the original \mathcal{H} .

By the above assumption, if P is a cycle, then $\bar{\epsilon}_1 \leq \bar{\epsilon}_0 < \infty$ (cf. (5)). And if P is an open path, then the growth of ϵ is bounded because of (E1). So we always finish with a finite ϵ ; let $\bar{\epsilon}$ and $\bar{\mathcal{H}}$ denote the resulting ϵ and honeycomb, respectively. We assert that $\eta_{\bar{\mathcal{H}}} < \eta_{\mathcal{H}}$. Clearly $\beta_{\bar{\mathcal{H}}} \leq \beta_{\mathcal{H}}$. Our further analysis relies on the following observations.

(i) When ϵ grows, each point u'_i uniformly moves along a ray from u_i , and each line L'_i uniformly moves in a normal direction to L_i . This implies that one can select a finite sequence $0 = \epsilon(0) < \epsilon(1) < \dots < \epsilon(N) = \bar{\epsilon}$ with $N = O(|\mathcal{V}|^2)$ such that for $t = 0, \dots, N - 1$, the honeycomb \mathcal{H}' does not change topologically when ϵ grows within the open interval $(\epsilon(t), \epsilon(t + 1))$.

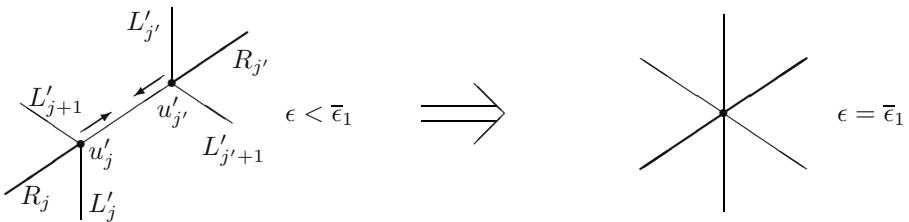
(ii) When ϵ starts growing from zero, each vertex v of \mathcal{H} occurring in P splits into several vertices whose total divergency is equal to the original divergency at v . By the construction of \mathcal{P}' and (iii) in Lemma 1, these vertices have the same sign. This implies that the total excess of these vertices is equal to the original excess at v . Each arising vertex u'_i has excess 1, which preserves during the process except possibly for those moments $\epsilon(t)$ when u'_i meets another vertex of \mathcal{H}' . When two or more vertices meet, their divergencies are added up. Therefore, the sum of their excesses (before the meeting) is strictly more than the resulting excess if some of these vertices have different signs. This implies that δ reduces if (E2) happens, or if ϵ reaches $\bar{\epsilon}_0$ (since the ends of each finite line L'_i have different signs, and the vertices u'_{i-1} and u'_i meet when L'_i vanishes). The latter situation is illustrated in the picture.



(iii) Let v be an integral vertex of the initial honeycomb \mathcal{H} and let W be the total weight of its incident edges in a current honeycomb \mathcal{H}' (depending on ϵ). If, at some moment, the point v is captured by the interior of some line L'_i , this increases W by 2. Now suppose some vertex u'_i meets v . If P turns right at u_i , then W increases by at least $w'(R_i) = 1$. And if P turns left at u_i , then the lines L'_i, L'_{i+1} do not vanish (cf. (5)), whence W increases by $w'(L'_i) + w'(L'_{i+1}) + w'(R_i) = 1$. Therefore, W increases when (E3) happens.

(iv) Let $\epsilon = \bar{\epsilon}_1 < \bar{\epsilon}_0$. By the maximality of $\bar{\epsilon}_1$ and reasonings in the proof of Lemma 2, a growth of ϵ beyond $\bar{\epsilon}_1$ would make some value $(w')_i^s(v)$ be negative. This can happen only in two cases: (a) some line R_j with weight -1 is covered

by edges of \mathcal{H} when $\epsilon = \bar{\epsilon}_1$, and not covered when $\epsilon > \bar{\epsilon}_1$; or (b) some $R_j, R_{j'}$ with weight -1 each lie on the same infinite line, the points u'_j and $u'_{j'}$ move toward each other when $\epsilon < \bar{\epsilon}_1$ and these points meet when ϵ reaches $\bar{\epsilon}_1$ (then $R_j, R_{j'}$ become overlapping for $\epsilon > \bar{\epsilon}_1$). One can see that, in case (a), u'_j meets a vertex v of \mathcal{H} (when $\epsilon = \bar{\epsilon}_1$) and the signs of v, u_j are different, and in case (b), the signs of $u_j, u_{j'}$ are different as well. In both cases, δ decreases (cf. (ii)). Case (b) is illustrated in the picture.



Using these observations, one can conclude that during the process β and δ are monotone nonincreasing and ω is monotone nondecreasing. Moreover, at least one of these values must change. Hence $\eta_{\overline{\mathcal{H}}} < \eta_{\mathcal{H}}$, as required. (We leave it to the reader to examine details more carefully where needed.)

Let \overline{h} be the concave cocirculation in G determined by $\overline{\mathcal{H}}$. (The graph G does not change as it is determined by the list of numbers $w(\mathcal{B}_i^s)$, defined in Section 2, and this list preserves.) To justify the induction and finish the proof of the theorem, it remains to explain that

$$I_h \subseteq I_{\overline{h}} \quad \text{and} \quad O_h \subseteq O_{\overline{h}}. \quad (6)$$

Denote by \mathcal{H}_ϵ and h_ϵ the current honeycomb \mathcal{H}' and the induced concave cocirculation h' at a moment ϵ in the process, respectively. The correspondence between the vertices of \mathcal{H}_ϵ and the flatspaces of h_ϵ (explained in Section 2) implies that for each edge $e \in E(G)$, the function $h_\epsilon(e)$ is continuous within each interval $(\epsilon(t), \epsilon(t+1))$ (cf. (i) in the above analysis). We assert that $h' = h_\epsilon$ is continuous in the entire segment $[0, \bar{\epsilon}]$ as well.

To see the latter, consider the honeycomb $\mathcal{H}_{\epsilon(t)}$ for $0 \leq t < N$. When ϵ starts growing from $\epsilon(t)$ (i.e., $\epsilon(t) < \epsilon < \epsilon(t+1)$ and $\epsilon - \epsilon(t)$ is small) the set $Q(v)$ of vertices of $\mathcal{H}' = \mathcal{H}_\epsilon$ arising from a vertex v of $\mathcal{H}_{\epsilon(t)}$ (by splitting or moving or preserving v) is located in a small neighbourhood of v . Moreover, for two distinct vertices u, v of $\mathcal{H}_{\epsilon(t)}$, the total weight of edges of \mathcal{H}_ϵ connecting $Q(u)$ and $Q(v)$ is equal to the weight of the edge between u and v in $\mathcal{H}_{\epsilon(t)}$ (which is zero when the edge does not exist), and all these edges are parallel. This implies that for each vertex v of $\mathcal{H}_{\epsilon(t)}$, the arising subgrids $G_{v'}, v' \in Q(v)$, in G (concerning h_ϵ) give a partition of the subgrid G_v (concerning $h_{\epsilon(t)}$), i.e., the set of little triangles of G contained in these $G_{v'}$ coincides with the set of little triangles occurring in G_v . So h' is continuous within $[\epsilon(t), \epsilon(t+1)]$.

Similarly, for each vertex v of $\mathcal{H}_{\epsilon(t)}$ ($0 < t \leq N$), the subgrid G_v is obtained by gluing together the subgrids $G_{v'}, v' \in Q'(v)$, where $Q'(v')$ is the set of vertices

of \mathcal{H}_ϵ (with $\epsilon(t-1) < \epsilon < \epsilon(t)$) that approach v when ϵ tends to $\epsilon(t)$. So h' is continuous globally. Now since no integral vertex of the initial honeycomb can move or split during the process (but merely merge with another vertex of \mathcal{H}' if (E3) happens), we conclude that the cocirculation preserves in all little triangles where it is integral initially, yielding the first inclusion in (6).

The second inclusion in (6) is shown in a similar fashion, relying on (E1).

This completes the proof of Theorem 2.

6 Polyhedra Having Vertices with Big Denominators

In this section we develop a construction proving Theorem 3. We start with some definitions and auxiliary statements.

Given a concave cocirculation h in a convex grid G , the *tiling* τ_h is the subdivision of the polygon $\mathcal{R}(G)$ spanned by G into the flatspaces T of h . We also call T a *tile* in τ_h . The following elementary property of tilings will be important for us.

Lemma 3. *Let $F \subseteq E(G)$, $h_0 : F \rightarrow \mathbb{R}$, and $h \in \mathcal{C}(G, h_0) =: \mathcal{C}$. Then h is a vertex of \mathcal{C} if and only if h is determined by its tiling, i.e., $h' \in \mathcal{C}$ and $\tau_{h'} = \tau_h$ imply $h' = h$.*

Proof. Since the polyhedron \mathcal{C} is the solution set of the system consisting of 3-circuit equalities and rhombus inequalities (RC), h is a vertex of \mathcal{C} if and only if it is determined by the set Q of little rhombi for which (RC) turns into equality (taking into account that if (RC) turns into equality for one pair of parallel edges of a rhombus ρ , then it does so for the other pair). Observe that each tile in τ_h one-to-one corresponds to a component of the graph whose vertices are the little triangles of G and whose edges are the pairs of little triangles forming rhombi in Q . This implies the lemma. \square

We will use a re-formulation of this lemma involving honeycombs. Let us say that honeycombs $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$ and $\mathcal{H}' = (\mathcal{V}', \mathcal{E}', w')$ are *conformable* if $|\mathcal{V}| = |\mathcal{V}'|$, $|\mathcal{E}| = |\mathcal{E}'|$, and there are bijections $\alpha : \mathcal{V} \rightarrow \mathcal{V}'$ and $\beta : \mathcal{E} \rightarrow \mathcal{E}'$ such that: for each vertex $v \in \mathcal{V}$ and each edge $e \in \mathcal{E}$ incident to v , the edge $\beta(e)$ is incident to $\alpha(v)$, $w(e) = w'(\beta(e))$, and if e is contained in $\Xi_i^s(v)$, then $\beta(e)$ is contained in $\Xi_i^s(\alpha(v))$. (This matches the situation when two concave cocirculations in G have the same tiling.)

Next, for $\mathcal{F} \subseteq \mathcal{E}$, we call \mathcal{H} *extreme* with respect to \mathcal{F} , or \mathcal{F} -*extreme*, if there is no honeycomb $\mathcal{H}' \neq \mathcal{H}$ such that \mathcal{H}' is conformable to \mathcal{H} and satisfies $d^c(\beta(e)) = d^c(e)$ for all $e \in \mathcal{F}$, where β is the corresponding bijection. Then the relationship between the tiling of concave cocirculations and the vertex sets of honeycombs leads to the following re-formulation of Lemma 3.

Corollary 1. *Let $F \subseteq E(G)$, $h_0 : F \rightarrow \mathbb{R}$, and $h \in \mathcal{C}(G, h_0)$. Let \mathcal{H} be the honeycomb determined by h and let \mathcal{F} be the subset of edges of \mathcal{H} corresponding to sides of tiles in τ_h that contain at least one edge from F . Then h is a vertex of $\mathcal{C}(G, h_0)$ if and only if \mathcal{H} is \mathcal{F} -extreme.*

One sufficient condition on extreme honeycombs will be used later. Let us say that a line ℓ in \mathbb{R}^2 is a *line of \mathcal{H}* if ℓ is covered by edges of \mathcal{H} . Then

- (C) \mathcal{H} is \mathcal{F} -extreme if each vertex of \mathcal{H} is contained in at least two maximal lines of \mathcal{H} , each containing an edge from \mathcal{F} .

Indeed, if two different maximal lines L, L' of \mathcal{H} intersect at a vertex v , then the constant coordinates of L, L' determine the dual coordinates of v . One can see that if \mathcal{H}' is conformable to \mathcal{H} , then the images of L, L' in \mathcal{H}' are maximal lines there and they intersect at the image of v . This easily implies (C).

The idea of our construction is as follows. Given $k \in \mathbb{N}$, we devise two honeycombs. The first honeycomb $\mathcal{H}' = (\mathcal{V}', \mathcal{E}', w')$ has the following properties:

- (P1) (i) the boundary $\mathcal{B}(\mathcal{H}')$ is partitioned into three sets $\mathcal{B}_1', \mathcal{B}_2', \mathcal{B}_3'$, where \mathcal{B}_i' consists of semiinfinite edges of the form $\Xi_i^+(\cdot)$, and $w'(\mathcal{B}_i') \leq Ck$, where C is a constant;
(ii) the constant coordinates of all edges of \mathcal{H}' are integral;
(iii) \mathcal{H}' is extreme with respect to $\mathcal{B}_1' \cup \mathcal{B}_2'$.

The second honeycomb $\mathcal{H}'' = (\mathcal{V}'', \mathcal{E}'', w'')$ has the following properties:

- (P2) (i) each semiinfinite edge of \mathcal{H}'' is contained in a line of \mathcal{H}' (in particular, $d^c(e)$ is an integer for each $e \in \mathcal{B}(\mathcal{H}'')$), and $w''(\mathcal{B}(\mathcal{H}'')) \leq w'(\mathcal{B}(\mathcal{H}'))$;
(ii) there is $e \in \mathcal{E}''$ such that the denominator of $d^c(e)$ is equal to k ;
(iii) \mathcal{H}' is extreme with respect to its boundary $\mathcal{B}(\mathcal{H}'')$.

Define the *sum* $\mathcal{H}' + \mathcal{H}''$ to be the honeycomb $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$ determined by the pre-honeycomb in which the line set is the (disjoint) union of \mathcal{E}' and \mathcal{E}'' , and the weight of a line e is equal to $w'(e)$ for $e \in \mathcal{E}'$, and $w''(e)$ for $e \in \mathcal{E}''$. Then each edge of \mathcal{H} is contained in an edge of \mathcal{H}' or \mathcal{H}'' , and conversely, each edge of \mathcal{H}' or \mathcal{H}'' is contained in a line of \mathcal{H} . Using this, one can derive from (P1) and (P2) that

- (P3) (i) the boundary $\mathcal{B}(\mathcal{H})$ is partitioned into three sets $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$, where \mathcal{B}_i consists of semiinfinite edges of the form $\Xi_i^+(\cdot)$, and $w(\mathcal{B}_i) \leq 2Ck$;
(ii) $d^c(e) \in \mathbb{Z}$ for all $e \in \mathcal{B}(\mathcal{H})$, and $d^c(e)$ has denominator k for some $e \in \mathcal{E}$;
(iii) \mathcal{H} is extreme with respect to $\mathcal{B}_1 \cup \mathcal{B}_2$.

(Property (iii) follows from (P1)(iii) and (P2)(i),(iii).)

Now consider the grid G and the concave cocirculation h in it determined by \mathcal{H} . By (P3)(i), G is a 3-side grid of size at most $2Ck$, with sides B_1, B_2, B_3 corresponding to $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$, respectively. Let $F := B_1 \cup B_2$ and let h_0 be the restriction of h to F (considering a side as edge set). Then (P3) together with Corollary 1 implies that G, h_0, h are as required in Theorem 3.

It remains to devise \mathcal{H}' and \mathcal{H}'' as above. To devise \mathcal{H}' is rather easy. It can be produced by truncating the dual grid formed by all lines with integer constant coordinates. More precisely, let n be a positive integer (it will depend on k). For

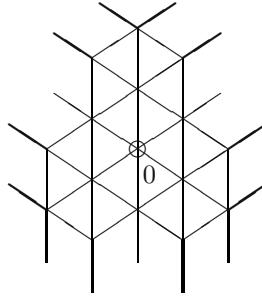
a point $x \in \mathbb{R}^2$ and $i = 1, 2, 3$, let x^i stand for the dual coordinate $d_i(x)$. The vertex set \mathcal{V}' consists of the points v such that

$$\begin{aligned} v^i &\in \mathbb{Z} \quad \text{and} \quad |v^i| < n, \quad i = 1, 2, 3, \\ v^1 - v^2 &\leq n, \quad v^2 - v^3 \leq n, \quad v^3 - v^1 \leq n. \end{aligned}$$

The finite edges of \mathcal{H}' have unit weights and connect the pairs $u, v \in \mathcal{V}'$ such that $|u^1 - v^1| + |u^2 - v^2| + |u^3 - v^3| = 2$. The semiinfinite edges and their weights are assigned as follows (taking indices modulo 3):

- if $v \in \mathcal{V}'$, $i \in \{1, 2, 3\}$, and $v^i - v^{i+1} \in \{n, n-1\}$, then \mathcal{H}' has edge $e = \Xi_{i-1}^+(v)$, and the weight of e is equal to 1 if $v^i - v^{i+1} = n-1$ and $v^i, v^{i+1} \neq 0$, and equal to 2 otherwise.

The case $n = 3$ is illustrated in the picture, where the edges with weight 2 are drawn in bold.



One can check that \mathcal{H}' is indeed a honeycomb and satisfies (P1)(i),(ii) (when $n = O(k)$). Also each vertex belongs to two lines of \mathcal{H}' , one containing a semiinfinite edge in \mathcal{B}'_1 , and the other in \mathcal{B}'_2 . Then \mathcal{H}' is $(\mathcal{B}'_1 \cup \mathcal{B}'_2)$ -expreme, by assertion (C). So \mathcal{H}' satisfies (P1)(iii) as well. Note that the set of semiinfinite edges of \mathcal{H}' is dense, in the sense that for each $i = 1, 2, 3$ and $d = -n+1, \dots, n-1$, there is a boundary edge e of the form $\Xi_i^+(\cdot)$ such that $d^c(e) = d$.

Next we have to devise \mathcal{H}'' satisfying (P2), which is less trivial. In order to facilitate the description and technical details, we go in reverse direction: we construct a certain concave cocirculation \tilde{h} in a convex grid \tilde{G} and then transform it into the desired honeycomb.

The grid \tilde{G} spans a hexagon with S- and N-sides of length 1 and with SW-, NW-, SE-, and NE-sides of length k . We denote the vertices in the big sides (in the order as they follow in the side-path) by:

- v1. x_k, x_{k-1}, \dots, x_0 for the SW-side;
- v2. $x'_k, x'_{k-1}, \dots, x'_0$ for the NW-side;
- v3. y_0, y_1, \dots, y_k for the SE-side;
- v4. y'_0, y'_1, \dots, y'_k for the NE-side.

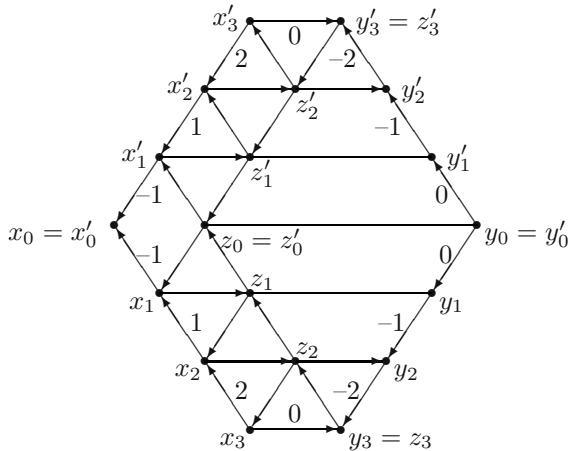
(Then $x_0 = x'_0$ and $y_0 = y'_0$.) We also distinguish the vertices $z_i := x_i + \xi_1$ and $z'_i := x'_i + \xi_1$ for $i = 1, \dots, k$ (then $z_0 = z'_0$, $z_k = y_k$, $z'_k = y'_k$.) We arrange an (abstract) tiling τ in \tilde{G} . It consists of

- t1. $2k - 2$ trapezoids and two little triangles obtained by subdividing the rhombus $R := y_k y_0 y'_k z_0$ (labeled via its vertices) by the horizontal lines passing $y_{k-1}, \dots, y_0, y'_1, \dots, y'_{k-1}$;
- t2. the little rhombus $\bar{\rho} := x_1 z_0 x'_1 x_0$;
- t3. $4k - 2$ little triangles $\Delta_i := x_i z_i z_{i-1}$, $\nabla'_i := x'_i z'_{i-1} z'_i$ for $i = 1, \dots, k$, and $\nabla_j := x_j x_{j+1} z_j$, $\Delta'_j := x'_j z'_j x'_{j+1}$ for $j = 1, \dots, k - 1$.

Define the function $h_0 : B(\tilde{G}) \rightarrow \mathbb{Z}$ by:

- h1. $h_0(x_i x_{i-1}) := h_0(x'_i x'_{i-1}) := i - 1$ for $i = 2, \dots, k$;
- h2. $h_0(x_1 x_0) := h_0(x'_1 x'_0) := -1$ and $h_0(x_k y_k) := h_0(x'_k y'_k) := 0$;
- h3. $h_0(y_{i-1} y_i) := h_0(y'_{i-1} y'_i) := -i + 1$ for $i = 1, \dots, k$.

The constructed τ and h_0 for $k = 3$ are illustrated in the picture (the numbers of the boundary edges indicate the values of h_0).



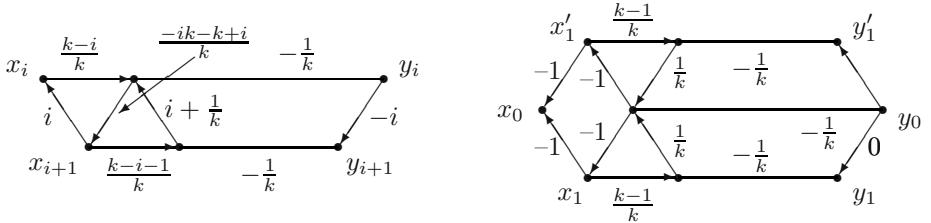
The tiling τ has the property that h_0 (as well as any function on $B(\tilde{G})$) is extendable to at most one cocirculation \tilde{h} in \tilde{G} such that \tilde{h} is *flat* within each tile T in τ , i.e., satisfies $\tilde{h}(e) = \tilde{h}(e')$ for any parallel edges e, e' in T . This is because we can compute \tilde{h} , step by step, using two observations: (i) the values of \tilde{h} on two nonparallel edges in a tile T determine \tilde{h} on all edges in T , and (ii) if a side S of a polygon P in \tilde{G} (spanned by a subgraph of \tilde{G}) is entirely contained in a tile, then the values of \tilde{h} on the edges of S are determined by the values on the other sides of P (since \tilde{h} is constant on S and $\tilde{h}(Q^c) = \tilde{h}(Q^a)$, where Q^c (Q^a) is the set of boundary edges of P oriented clockwise (anticlockwise) around P).

We assign \tilde{h} as follows. First assign $\tilde{h}(e) := h_0(e)$ for all $e \in B(\tilde{G})$. Second assign $\tilde{h}(z_0 x_1) := \tilde{h}(z_0 x'_1) := -1$ (by (i) for the rhombus $\bar{\rho}$ in τ). Third, for an edge e in the horizontal line $z_0 y_0$, assign

$$\begin{aligned}\tilde{h}(e) &:= \frac{1}{k} \left(\tilde{h}(z_0 x_1) - \tilde{h}(x_k x_1) + \tilde{h}(x_k y_k) - \tilde{h}(y_0 y_k) \right) \\ &= \frac{1}{k} \left(-1 - (1 + \dots + (k-1)) + 0 - (0 - 1 - \dots - (k-1)) \right) = -1/k\end{aligned}$$

(by (ii) for the pentagon $z_0x_1x_ky_ky_0$, taking into account that the side z_0y_0 contains k edges), where $\tilde{h}(uv)$ is the sum of values of \tilde{h} on the edges of a side uv . Therefore, $\tilde{h}(e) = -1/k$ for all horizontal edges in the big rhombus R , and we then can assign $\tilde{h}(z_{i+1}z_i) := \tilde{h}(z'_{i+1}z'_i) := i + 1/k$ for $i = 0, \dots, k-1$ (by applying (i) to the trapezoids and little triangles of τ in R). Fourth, repeatedly applying (i) to the little triangles $\Delta_1, \nabla_1, \Delta_2, \dots, \nabla_{k-1}, \Delta_k$ (in this order) that form the trapezoid $x_1x_ky_kz_0$ in which \tilde{h} is already known on all side edges, we determine \tilde{h} on all edges in this trapezoid; and similarly for the symmetric trapezoid $z'_0y'_kx'_kx'_1$.

One can check that the obtained cocirculation \tilde{h} is well-defined, and moreover, it is a concave cocirculation with tiling τ (a careful examination of the corresponding rhombus inequalities is left to the reader). Since \tilde{h} is computed uniquely in the process, it is a vertex of $\mathcal{C}(\tilde{G}, h_0)$. Also \tilde{h} is integral on the boundary of \tilde{G} , has an entry with denominator k , and satisfies $|\tilde{h}(e)| < 2k$ for all $e \in E(\tilde{G})$. The picture indicates the values of \tilde{h} in the horizontal strip between the lines x_iy_i and $x_{i+1}y_{i+1}$ for $1 \leq i \leq k-2$, and in the horizontal strip between x_1y_1 and $x'_1y'_1$.



Let $\tilde{\mathcal{H}}$ be the honeycomb determined by (\tilde{G}, \tilde{h}) . It is $\mathcal{B}(\tilde{\mathcal{H}})$ -extreme, by Corollary 1, has all boundary edges integral and has a finite edge with constant coordinate $1/k$. We slightly modify $\tilde{\mathcal{H}}$ in order to get rid of the semiinfinite edges e of the form $\Xi_i^-(v)$ in it. This is done by truncating such an e to finite edge $e' = vu$ and adding two semiinfinite edges $a := \Xi_{i-1}^+(u)$ and $b := \Xi_{i+1}^+(u)$, all with the weight equal to that of e (which is 1 in our case). Here u is the integer point in $\Xi_i^-(v) \setminus \{v\}$ closest to v . (Strictly speaking, when applying this operation simultaneously to all such edges e , we should handle the corresponding pre-honeycomb, as some added edge may intersect another edge at an interior point.) The semiinfinite edges e of the obtained honeycomb $\mathcal{H}'' = (\mathcal{V}'', \mathcal{E}'', w'')$ are of the form $\Xi_i^+(\cdot)$, and $d^c(e)$ is an integer between $-2k$ and $2k$. Also $w''(\mathcal{B}(\mathcal{H}'')) < 2|B(\tilde{G})|$. This honeycomb is extreme with respect to its boundary since so is $\tilde{\mathcal{H}}$ and since for e', a, b as above, the constant coordinate of e' is determined by the constant coordinates of a, b . Thus, \mathcal{H}'' satisfies (P2) when $n > 2k$ (to ensure that each semiinfinite edge of \mathcal{H}'' is contained in a line of \mathcal{H}').

Now (G, h) determined by $\mathcal{H}' + \mathcal{H}''$ is as required in Theorem 3. \square

Remark 2. In our construction of vertex \tilde{h} of $\mathcal{C}(\tilde{G}, h_0)$, the design of tiling τ is borrowed from one fragment in a construction in [2] where one shows that (in

our terms) the polytope of semiconcave cocirculations in a 3-side grid with fixed integer values on the boundary can have a vertex with denominator k . Here we call a cocirculation h *semiconcave* if (RC) holds for each rhombus ρ whose diagonal edge d is parallel to ξ_2 or ξ_3 (but may be violated if d is parallel to ξ_1).

Remark 3. The first honeycomb \mathcal{H}' in our construction turns out to be \mathcal{F}' -extreme for many proper subsets \mathcal{F}' of $\mathcal{B}(\mathcal{H}')$ and even of $\mathcal{B}'_1 \cup \mathcal{B}'_2$. For example, one can take $\mathcal{F}' := \mathcal{B}'_1 \cup \{e\}$, where e is an arbitrary edge in \mathcal{B}'_2 (a check-up is left to the reader). Moreover, it is not difficult to produce certain triples of boundary edges that possess such a property. For each of these sets \mathcal{F}' , the honeycomb $\mathcal{H} = \mathcal{H}' + \mathcal{H}''$ is \mathcal{F} -extreme, where \mathcal{F} consists of the semiinfinite edges of \mathcal{H} contained in members of \mathcal{F}' . Then, by Corollary 1, the constructed concave cocirculation h is a vertex of the polyhedron $\mathcal{C}(G, h|_F)$ as well, where F is a subset of boundary edges of G whose images in \mathcal{H} cover \mathcal{F} (i.e., for each edge $\ell \in \mathcal{F}$ of the form $\Xi_i^+(\cdot)$, there is an $e \in F \cap B_i$ with $h(e) = d^e(\ell)$). This gives a strengthening of Theorem 3.

Remark 4. It seems to be more intricate to describe the above construction implying Theorem 3 so as to use only the language of concave cocirculations. In particular, the operation on a pair h', h'' of concave cocirculations analogous to taking the sum of honeycombs $\mathcal{H}', \mathcal{H}''$ is less transparent (it is related to taking the convolution of concave functions behind h', h''). This is why we prefer to argue in terms of honeycombs.

7 Concluding Remarks

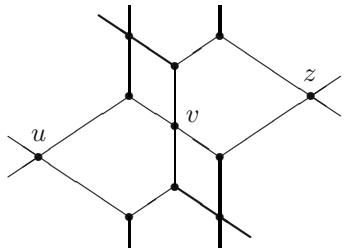
The proof of Theorem 2 in Section 5 provides a strongly polynomial algorithm which, given G and h , finds h' as required in this theorem. Here the number of iterations (viz. applications of the induction on η) is $O(n)$, where $n := |E(G)|$. The number of steps at an iteration (viz. moments ϵ when some line L'_i captures a vertex of \mathcal{H} or when two vertices u'_i, u'_j meet) is $O(n^2)$, and these moments can be computed easily. To find $\bar{\epsilon}_1$ is easy as well. Hence an iteration is performed in time polynomial in n .

As a consequence, we have a strongly polynomial algorithm to solve the following problem: given a convex grid G and a function $h_0 : B(G) \rightarrow \mathbb{Z}$, decide whether h_0 is extendable to a concave cocirculation in G , and if so, find an integer concave cocirculation h with $h|_{B(G)} = h_0$. This is because the problem of finding a concave cocirculation having prescribed values on the boundary can be written as a linear program of size $O(n)$.

One can slightly modify the method of proof of Theorem 2 so as to obtain the following strengthening: for a concave cocirculation h in a convex grid G , there exists an integer concave cocirculation h' satisfying $h'(e) = h(e)$ for each edge $e \in O_h \cup I'_h$, where I'_h is the set of edges contained in circuits C of G such that h takes integer values on all edges of edges C . We omit the proof here.

Next, as mentioned in the Introduction, a concave cocirculation h in a convex grid G need not admit an improvement to an integer concave cocirculation in G preserving the values on *all* edges where h is integer. (Note that in our proof of Theorem 2, a vertex of the original honeycomb having only one integer dual coordinate may create vertices not preserving this coordinate.) A counterexample (G, h) is shown in the picture (the right figure illustrates the corresponding honeycomb; here all edge weights are ones, the integral edges are drawn in bold, and each of the vertices u, v, z has one integer dual coordinate).

$$\begin{array}{ccccccc} & \bullet & 1 & \bullet & 0 & \bullet & \\ & 0 & -1 & 0 & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \bullet & 1 & \bullet & \frac{1}{2} & \bullet & -1 & \bullet \\ -\frac{3}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \bullet & 2 & \bullet & \frac{1}{2} & \bullet & 0 & \bullet \\ -\frac{1}{2} & -\frac{3}{2} & \frac{1}{2} & -1 & 1 & -1 & \\ \bullet & 1 & \bullet & 0 & \bullet & & \end{array}$$



One can check that this h is determined by its integer values, i.e., $\mathcal{C}(G, h|_F) = \{h\}$, where F is the set of edges where h is integer.

We finish with the following question motivated by some aspects in Section 6. For a convex grid G , can one give a “combinatorial characterization” for the set of tilings of concave cocirculations h such that h is a vertex of the polytope $\mathcal{C}(G, h|_{B(G)})$?

Acknowledgement. I thank Vladimir Danilov for stimulating discussions and Gleb Koshevoy for pointing out to me a polyhedral result in [2].

References

1. A.S. Buch, The saturation conjecture (after A. Knutson and T. Tao). With an appendix by William Fulton, *Enseign. Math. (2)* **46** (2000), no. 1-2, 43–60.
2. J.A. De Loera and T.B. McAllister, Vertices of Gelfand-Tsetlin polytopes, *arXiv:math.CO/0309329*, 2003.
3. A.V. Karzanov, Concave cocirculations in a triangular grid, submitted to *Linear Algebra and Appl.*
4. A. Knutson and T. Tao, The honeycomb model of $GL_n(\mathbb{C})$ tensor products I: Proof of the saturation conjecture, *J. Amer. Math. Soc.* **12** (4) (1999) 1055–1090.
5. A. Knutson, T. Tao, and C. Woodward, The honeycomb model of $GL_n(\mathbb{C})$ tensor products II: Puzzles determine facets of the Littlewood-Richardson cone, *Preprint*, 2001; to appear in *J. Amer. Math. Soc.*

Minsquare Factors and Maxfix Covers of Graphs

Nicola Apollonio^{1*} and András Sebő²

¹ Department of Statistics, Probability, and Applied Statistics University of Rome
“La Sapienza”, Rome, Italy

² CNRS, Leibniz-IMAG, Grenoble, France

Abstract. We provide a polynomial algorithm that determines for any given undirected graph, positive integer k and various objective functions on the edges or on the degree sequences, as input, k edges that minimize the given objective function. The tractable objective functions include linear, sum of squares, etc. The source of our motivation and at the same time our main application is a subset of k vertices in a line graph, that cover as many edges as possible (maxfix cover). Besides the general algorithm and connections to other problems, the extension of the usual improving paths for graph factors could be interesting in itself: the objects that take the role of the improving walks for b -matchings or other general factorization problems turn out to be edge-disjoint unions of pairs of alternating walks. The algorithm we suggest also works if for any subset of vertices upper, lower bound constraints or parity constraints are given. In particular maximum (or minimum) weight b -matchings of given size can be determined in polynomial time, combinatorially, in more than one way.

1 Introduction

Let $G = (V, E)$ be a graph that may contain loops and parallel edges, and let $k > 0$ be an integer. The main result of this work is to provide a polynomial algorithm for finding a subgraph of cardinality k that minimizes some pre-given objective function on the edges or the degree sequences of the graph. The main example will be the sum of the squares of the degrees (minsquare problem) showing how the algorithm works for the sum of any one dimensional convex function of the degrees (Section 3.1), including also linear functions (Section 3.2). The sum of squares function is general enough to exhibit the method in full generality, and at the same time concrete enough to facilitate understanding, moreover this was originally the concrete problem we wanted to solve. It also arises in a natural way in the context of vertex-covers of graphs, and this was our starting point:

Given a graph and an integer t find a subset of vertices of cardinality t that cover the most number of edges in a graph, that is find a *maxfix cover*. This problem, introduced by Petrank in [14] under the name of *max vertex cover*, obviously

* On leave from Università di Roma “La Sapienza”, postdoc fellowship supported by the European project DONET while the author visited Laboratoire Leibniz (IMAG)

contains the vertex cover problem, so it is NP-hard in general. However, VERTEX COVER is polynomially solvable for line graphs (it is straightforwardly equivalent to the maximum matching problem). What about the maxfix cover problem in line graphs?

A maxfix cover for $L(G)$ is $T \subseteq E(G)$ minimizing the number of incident pairs of edges in the remaining edge-set $F = E(G) \setminus T$, $|F| = k := n - t$. Clearly, the number of such paths is $\sum_{v \in V} \binom{d_F(v)}{2}$. Since the sum of the degrees is constant, this is equivalent to minimizing $\sum_{v \in V} d_F^2(v)$. This sum will be called the value of F . A subgraph of k edges will be called *optimal* if it minimizes the value, and the problem of finding an optimal subgraph of k edges will be called the *minsquare* problem. The main result of this work is to provide a polynomial algorithm for solving this problem.

Let us introduce some notation and terminology used throughout the paper. Let G be a graph. Then $n := n(G) := |V(G)|$; $E(X)$ ($X \subseteq V(G)$) is the set of edges induced by X , that is, with both endpoints in X ; $\delta(X)$ denotes the set of edges with exactly one endpoint in X . For $X \subseteq V(G)$ let $d(X) := |\delta(X)|$. We will not distinguish subgraphs from subsets of edges. For a subgraph $F \subseteq E(G)$ let $d_F(v)$ ($v \in V$) be the degree of v in F , that is, the number of edges of F incident to v . The maximum degree of G will be denoted by Δ_G . The line graph of G will be denoted by $L(G)$. The Euclidean norm of a vector $a \in \mathbb{R}^n$, denoted by $\|a\|$, is the number $\sqrt{\sum_{i=1}^n a_i^2}$ (thus $\|a\|^2 = \sum_{i=1}^n a_i^2$). The l_1 norm of a , denoted by $|a|$, is the number $|a| := \sum_{i=1}^n |a_i|$.

Given $b : V(G) \rightarrow \mathbb{N}$, a *b -matching* is a subset of edges $F \subseteq E(G)$ such that $d_F(v) = b(v)$ for every $v \in V(G)$; b is a *degree sequence* (in G) if there exists a b -matching in G . More generally, an (f, g) -factor, where $f, g : V(G) \rightarrow \mathbb{N}$, is $F \subseteq E(G)$ with $f(v) \geq d_F(v) \geq g(v)$ for all $v \in V(G)$.

In the same way as minimum vertex covers are exactly the complementary sets of maximum stable sets, maxfix covers are the complementary sets of ‘minfix induced subgraphs’, that is, of sets of vertices of pre-given cardinality that induce the less possible edges. (Ad extrema 0 edges, when the decision version of the minfix induced subgraph problem with input k specializes to answering the question ‘is $\alpha \geq k$?’.) Similarly, minfix covers are the complements of maxfix induced subgraphs.

As we will see in 3.4, among all these variants the only problem that can be solved in relatively general cases is maxfix cover. The others are NP-hard already in quite special cases.

The maxfix cover problem has been even more generally studied, for hypergraphs: find a set of vertices of given size $t \in \mathbb{N}$ that hits the most number (highest weight) of hyperedges. For *Edge-Path* hypergraphs, that is a hypergraphs whose vertices are the edges of a given underlying graph G and whose set of hyperedges is a given family of weighted paths in G , several results have been achieved:

In Apollonio et al. in [1] and [2] polynomial algorithms have been worked out for special underlying graphs (caterpillars, rooted arborescences, rectangular grids with a fixed number of rows, etc.) and for special shaped collection of paths

(staple-free, rooted directed paths, L -shaped paths, etc.), and it has been shown that the problem is NP-complete for a fairly large set of special Edge-Path hypergraphs. When the Edge-Path hypergraph has the form (G, \mathcal{P}) , \mathcal{P} being the family of all paths of length two in G , the problem is a maxfix cover problem in the line graph $L(G)$ of G .

Until the last section we will state the results in terms of the minsquare problem, because of the above mentioned application, and the general usability and intuitive value of the arguments.

A support for the polynomial solvability of the minsquare problem is general convex optimization. It is well-known [9] that convex function can be minimized in polynomial time on convex polytopes under natural and general conditions. Hence convex functions can be optimized on ‘ b -matching polytopes’ and intersections of such polytopes with hyperplanes (or any other solvable polyhedron in the sense of [9]). However, the optima are not necessarily integer, neither when minimizing on the polytope itself, nor for the intersection.

Minimizing a convex function on b -matchings, that is the integer points of the b -matching polytope, is still easy with standard tools: single improving paths suffice, and the classical algorithms for finding such paths [8], [15] do the job. However, for our problem, where the set of b -matchings is intersected with a hyperplane, single paths do no more suffice (see at the end of this Introduction); yet we will show that pairs of paths along which a non-optimal solution can be improved do always exist and yield a polynomial algorithm. In this way the integer optimum of a quite general convex function can still be determined in polynomial time on the intersection of (f, g) -factor polyhedra with hyperplanes. This is less surprising in view of the following considerations.

Intuitively, the best solution is the ‘less extremal’ one. Clearly, if $r := 2k/n$ is an integer and G has an r -regular subgraph, then it is an optimal solution of the minsquare problem. This is the ‘absolute minimum’ in terms of k and n . The existence of an r -regular subgraph is polynomially decidable (with the above mentioned methods) which makes the problem look already hopeful: it can be decided in polynomial time whether this absolute minimum can be attained or not.

If $2k/n$ is not integer, it is also clear to be uniquely determined how many edges must have degree $\lceil 2k/n \rceil$ and how many $\lfloor 2k/n \rfloor$ in a subgraph, so as the sum of the degrees of the subgraph is $2k$. However, now it is less straightforward to decide whether this absolute minimum can be attained or not, since the number of all cases to check may be exponential. At first sight the general problem may appear hopeless.

Yet the main result of the paper states that a subgraph F is optimal *if and only if there is no vector $t : V(G) \rightarrow \mathbb{N}$ such that:*

- t is a degree sequence in G ;
- $\sum_{v \in V} d_F(v) = \sum_{v \in V} t(v)$, that is each t -matching has the same size as F ;
- $\sum_{v \in V} |d_F(v) - t(v)| \leq 4$, that is t differs from F by at most 4 in l_1 -norm;
- $\sum_{v \in V} t^2(v) < \sum_{v \in V} d_F^2(v)$, that is, t has better objective value than F .

Since the number of vectors v that satisfy the last three conditions is smaller than n^4 , and it can be decided for each whether it satisfies the first condition by classical results of Tutte, Edmonds-Johnson, and various other methods (see accounts in [8], [15], [12]), the result implies a polynomial algorithm for the minsquare problem.

If t satisfies the above four conditions, the function (vector) $\kappa := t - d_F$ will be called an *improving* vector with respect to F . We have just checked:

(1) *If an improving vector exists it can be found in polynomial time.*

The graph G consisting of two vertex-disjoint triangles shows that one cannot replace 4 by 2 in the second condition, unlike in most of the other factorization problems. Indeed, choose $k = 4$, and let F contain the three edges of one of the triangles and one edge from the other. The value of this solution is 14, the optimum is 12 and one has to change the degree of at least four vertices to improve. Optimizing linear functions over the degree sequences of subgraphs of fixed cardinality k presents already the difficulties of the general case (see Section 3.2); on the other hand the methods apply for a quite general set of objective functions (see Section 3). We have chosen to put in the center minsquare factors because of the origins of the problem and because they are a representative example of the new problems that can be solved.

The rest of the paper is organized as follows: in Section 2 we develop the key lemmas that are behind the main result and make the algorithm work. Then in Section 2.2 we prove the main result and state a polynomial algorithm that solves the minsquare problem. In Section 3 we characterize the functions for which the procedure is valid, exhibit some additional conditions for which the method works, and state some connections to other problems.

2 Main Results

The following result is a variant of theorems about improving alternating walks concerning b -matchings (f -factors). In this paper we avoid speaking about refined details of these walks. We adopt a viewpoint that is better suited for our purposes, and focuses on *degree sequences*. (In Section 3.2 we mention some ideas concerning efficient implementation.)

Let G be a graph, and $F, F' \subseteq E(G)$. Then $P \subseteq E(G)$ will be called an $F - F'$ alternating walk, if $P \subseteq F \cup F'$ and $\sum_{v \in V} |d_{P \cap F}(v) - d_{P \cap F'}(v)| \leq 2$; even if $|\sum_{v \in V} d_{P \cap F}(v) - d_{P \cap F'}(v)| = 0$, odd if $|\sum_{v \in V} d_{P \cap F}(v) - d_{P \cap F'}(v)| = 2$. Clearly, an even walk contains the same number of edges of F and F' , and in an odd walk one of them has one more edge than the other.

An $F - E(G) \setminus F$ -alternating walk that has at least as many edges in F as in $E(G) \setminus F$ will be simply called an F -walk. For an F -walk P (where F is fixed) define $\kappa_P : V(G) \rightarrow \mathbb{Z}$ by $\kappa_P(v) := d_{P \setminus F}(v) - d_{P \cap F}(v)$, $v \in V$; clearly, $|\kappa_P| = 2$ or 0; κ_P will be called the *change* (of F along P).

2.1 The Key-Facts

We state here three simple but crucial facts:

- (2) *If $F \subseteq E(G)$ and P is an F -alternating walk, then $d_F + \kappa_P$ is a degree sequence.*

Indeed, $d_F + \kappa_P$ is the degree sequence of $F\Delta P$, where Δ denotes the symmetric difference, $F\Delta P := (F \setminus P) \cup (P \setminus F)$.

In other words, an alternating walk is a subgraph P of G that has the property that $d_{P \cap F} = d_{P \cap F'}$ in all but at most two vertices of G . The degree of F and F' can differ in two vertices (by 1) or in one vertex (by 2). We call these vertices the *endpoints* of the alternating walk, and if the two endpoints coincide we say that the vertex is an endpoint of the path with *multiplicity* 2 (twice).

Note that we will not use any fact or intuition about how these paths ‘go’, the only thing that matters is the change vector $\kappa_P := (d_{P \cap F'}(v) - d_{P \cap F}(v))_{v \in V}$, and the fact (2) about it: adding this vector to the degree sequences of F , we get a feasible degree sequence again.

If $|d_{P \cap F}(v) - d_{P \cap F'}(v)| = 0$ for all $v \in V$, that is in every node of P there is the same number of incident edges in F and F' , then we say it is an *alternating cycle*.

The following statement is a variant of folklore statements about improving paths concerning graph factors, generalizing Berge’s improving paths for matchings:

- (3) *Let $F, F' \subseteq E$. Then $F\Delta F'$ is the disjoint union of alternating walks, so that for all $v \in V$, v is the endpoint of $|d_F(v) - d_{F'}(v)|$ of them (with multiplicity).*

Equivalently, for every $v \in V$, the alternating walks in (3) starting at v either all start with an F -edge or all start with an F' -edge.

Indeed, to prove (3) note that $F\Delta F'$, like any set of edges, can be decomposed into edge-disjoint alternating walks: edges, as one element sets are alternating walks, and they are edge-disjoint. Take a decomposition that consists of a minimum number of walks. Suppose for a contradiction that for some $u \in V(G)$ there exist two walks, P_1 and P_2 such that $P_1 \cap F$ has more edges in u than $P_1 \cap F'$, $P_2 \cap F$ has less edges in u than $P_2 \cap F'$, and let $P := P_1 \cup P_2$. It follows that u is an endpoint of both P_1 and P_2 , moreover with different signs, and we get:

$$\begin{aligned} \sum_{v \in V} |d_{P \cap F}(v) - d_{P \cap F'}(v)| &\leq \sum_{v \in V} |d_{P_1 \cap F}(v) - d_{P_1 \cap F'}(v)| + \\ &\quad + \sum_{v \in V} |d_{P_2 \cap F}(v) - d_{P_2 \cap F'}(v)| - 2 \leq 2 + 2 - 2 = 2 . \end{aligned}$$

Therefore, P is also an alternating walk, hence it can replace $\{P_1, P_2\}$ in contradiction with the minimum choice of our decomposition, and (3) is proved. \square

We see that in case $|F| = |F'|$, the number of alternating walks in (3) with one more edge in F is the same as the number of those with one more edge in

F' . It follows that the symmetric difference of two subgraphs of the same size can be partitioned into edge-sets each of which consists either of an alternating path (also allowing circuits) or of the union of two (odd) alternating paths.

The statement (3) will be useful, since walks will turn out to be algorithmically tractable. For their use we need to decompose improving steps into improving steps on walks.

Let a and λ be given positive integers, and let $\delta(a, \lambda) := (a + \lambda)^2 - a^2 = 2\lambda a + \lambda^2$. Then we have:

If λ_1 and λ_2 have the same sign, then $\delta(a, \lambda_1 + \lambda_2) \geq \delta(a, \lambda_1) + \delta(a, \lambda_2)$.

For each given factor F and each given $\lambda \in \mathbb{Z}^{V(G)}$ define $\delta(F, \lambda)$ as $\|d_F + \lambda\|^2 - \|d_F\|^2$, that is,

$$\delta(F, \lambda) = \sum_{v \in V(G)} \delta(d_F(v), \lambda(v)) .$$

(4) If $\lambda_1, \dots, \lambda_t$ are vectors such that for every $v \in V(G)$, $\lambda_1(v), \dots, \lambda_t(v)$ have the same sign (this sign may be different for different v) and $\lambda = \lambda_1 + \dots + \lambda_t$, then $\delta(F, \lambda) \geq \delta(F, \lambda_1) + \dots + \delta(F, \lambda_t)$.

Indeed, apply the inequality stated above to every $v \in V$, and then sum up the n inequalities we got. \square

Now if F is not optimal, then by (3) and (4) one can also improve along pairs of walks. The details are worked out in the next section.

2.2 Solving the Minsquare Problem

Recall that for given $F \subseteq E(G)$ an improving vector is a vector $\kappa : V(G) \rightarrow \mathbb{Z}$ such that $b := d_F + \kappa$ is a degree sequence, $\sum_{v \in V(G)} |\kappa(v)| \leq 4$, and $\sum_{v \in V} b(v)^2 < \sum_{v \in V} d_F(v)^2$, while $\sum_{v \in V} d_F(v) = \sum_{v \in V} b(v)$.

Theorem 1. Let G be a graph. If a factor F is not optimal, then there exists an improving vector.

Proof. Let F_0 be optimal. As F is not optimal one has

$$0 > \|d_{F_0}\|^2 - \|d_F\|^2 = \|d_F + d_{F_0} - d_F\|^2 - \|d_F\|^2 = \delta(F, d_{F_0} - d_F) .$$

By (3) $F \Delta F_0$ is the disjoint union of $m \in \mathbb{N}$ F -alternating paths P_1, \dots, P_m . In other words, $F_0 = F \Delta P_1 \Delta \dots \Delta P_m$, and using the simplification $\kappa_i := \kappa_{P_i}$ we have:

$$d_{F_0} = d_F + \sum_{i=1}^m \kappa_i ,$$

where we know that the sum of the absolute values of coordinates of each κ_i ($i = 1, \dots, m$) is ± 2 or 0. Since F and F_0 have the same sum of coordinates

$$|\{i \in \{1, \dots, m\} : \sum_{v \in V(G)} \kappa_i(v) = 2\}| = |\{i \in \{1, \dots, m\} : \sum_{v \in V(G)} \kappa_i(v) = -2\}|,$$

and denote this cardinality by p .

Therefore those $i \in \{1, \dots, m\}$ for which the coordinate sum of κ_i is 2 can be perfectly coupled with those whose coordinate sum is -2 ; do this coupling arbitrarily, and let the sum of the two members of the couples be $\kappa'_1, \dots, \kappa'_p$. Clearly, for each κ'_i ($i = 1, \dots, p$) the coordinate-sum is 0, $\sum_{v \in V(G)} |\kappa'_i(v)| \leq 4$, and

$$d_{F_0} = d_F + \sum_{i=1}^p \kappa'_i .$$

Now by (2) each of $d_F + \kappa'_i$ ($i = 1, \dots, p$) is a degree sequence. To finish the proof we need that at least one of these is an improving vector, which follows from (4):

$$0 > \delta(F, d_{F_0} - d_F) = \delta(F, \sum_{i=1}^p \kappa'_i) \geq \sum_{i=1}^p \delta(F, \kappa'_i) .$$

It follows that there exists an index i , $1 \leq i \leq p$ such that $\delta(F, \kappa'_i) < 0$. \square

Corollary 2. *The minsquare and the maxfix-cover problem can be solved in polynomial time.*

Proof. Indeed, the maxfix cover problem has already been reduced (see beginning of the introduction) to the minsquare problem. Since the value of any solution, including the starting value of the algorithm, is at most n^3 , and an $O(n^3)$ algorithm applied n^4 times decreases it at least by 1, the optimum can be found in at most $O(n^{10})$ time. \square

It can be easily shown that the improving vectors provided by the theorem are in fact alternating walks - similarly to other factorization problems - or edge disjoint unions of such alternating walks. If someone really wants to solve such problems these paths can be found more easily (by growing trees and shrinking blossoms) than running a complete algorithm that finds a b -matching. By adding an extra vertex, instead of trying out all the n^4 possibilities, one matching-equivalent algorithm is sufficient for improving by one. However, the goal of this paper is merely to prove polynomial solvability. Some remarks on more refined methods can be found in 3.2.

Various polynomial algorithms are known for testing whether a given function $b : V \rightarrow \mathbb{N}$ is a degree sequence of the given graph $G = (V, E)$. Such algorithms are variants or extensions of Edmonds' algorithm for 1-matchings [6], and have been announced in [7]. The same problem can also be reduced to matchings. A variety of methods for handling these problems can be found in [12], [8], [15].

The complexity of a variant of the matching algorithm is bounded by $O(n^{2.5})$, and can be used for making an improving step; n^3 calls for this algorithm are sufficient, so if we are a bit more careful, $O(n^{5.5})$ operations are sufficient to find a minsquare factor.

3 Special Cases and Extensions

3.1 Characterizing when It Works

We will see here that the proof of Theorem 1 works if we replace squares by any set of functions $f_v : \mathbb{N} \rightarrow \mathbb{R}$ ($v \in V$) for which $\delta(F, \lambda) := \sum_{v \in V} f_v(d_F(v) + \lambda) - f_v(d_F(v))$ satisfies (4). This is just a question of checking. However, a real new difficulty arises for proving Corollary 2: the difference between the initial function value and the optimum is no more necessarily bounded by a polynomial of the input, it is therefore no more sufficient to improve the objective value by 1 in polynomial time.

The problem already arises for linear functions: suppose we are given rational numbers p_v ($v \in V$) on the vertices, and $f_v(x) := p_v x$. The input is $O(\log \max\{|p_v| : v \in V\})$, but if we cannot make sure a bigger improvement than by a constant, then we may need $O(\max\{|p_v| : v \in V\})$ steps.

However, this is a standard problem and has a standard solution, since a slight sharpening of (1) is true: the improving vector κ with the highest $\delta(F, \kappa)$ value can also be found in polynomial time. Indeed, one has to take the optimum of a polynomial number of values. Along with the following standard trick the polynomial bound for the length of an algorithm minimizing $\sum_{v \in V} f_v(d_F(v))$ among subgraphs $F \subseteq E(G)$ can be achieved:

(5) *Starting with an arbitrary F_0 and choosing repeatedly an improving vector κ with maximum $|\delta(F, \kappa)|$, that is, minimum $\delta(F, \kappa) < 0$ value, there are at most $O(n^2 \log \max\{|p_v| : v \in V\})$ improving steps.*

Note that the case of linear functions that we use for an example can be solved very easily independently of our results. It is a special case of problems minimizing a linear function on the edges, that is of the following problem: given $w : E(G) \rightarrow \mathbb{Z}$ and $k \in \mathbb{N}$ minimize the sum of the edge-weights among subgraphs of cardinality k . (The node-weighted problem can be reduced to edge-weights defined with $w(ab) := p_a + p_b$ ($a, b \in V$); indeed, then $w(F) = \sum_{v \in V} p_v d_F(v)$.) Add now an extra vertex x_0 to the graph and join it with every $v \in V(G)$ by $d_G(v)$ parallel edges. A minimum weight subgraph with degrees equal to $2(|E(G)| - k)$ in x_0 and $d_G(v)$ for all $v \in V$ intersects $E(G)$ in a minimum weight k -cardinality subgraph. (The same can be achieved under more constraints see 3.2.)

Let us make clear now the relation of inequality (4) with some well-known notions.

A function $f : D \rightarrow \mathbb{R}$ ($D \subseteq \mathbb{R}^n$) is said to be convex if for any $x, y \in D$ and $\alpha, \beta \in \mathbb{R}$, $\alpha + \beta = 1$ such that $\alpha x + \beta y \in D$ we have $f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y)$.

Note that we do not require D to be convex, that is for instance D can also be any subset of integers.

A particular case of the defining inequality that will actually turn out to be equivalent is

$$(f(x_1) + f(x_2))/2 \geq f((x_1 + x_2)/2),$$

that is, say for $x_1 = a$, $x_2 = a + 2$:

$$f(a) + f(a + 2) \geq 2f(a + 1),$$

that is,

$$f(a + 2) - f(a + 1) \geq f(a + 1) - f(a).$$

We are not surprised to get this inequality, which characterizes supermodularity, strictly related to discrete convexity, see Murota's work [13]. Let us state the equivalence of these inequalities in a form useful for us:

Lemma 3. *The following statements are equivalent about the function f whose domain is a (possibly infinite) interval:*

- (i) f is convex
- (ii) For every integer i so that $i, i-1, i+1 \in D(f)$: $f(i) \leq (f(i-1) + f(i+1))/2$.
- (iii) If $x = x_1 + x_2$, where x_1, x_2 have the same sign, then $f(a + x) - f(a) \geq f(a + x_1) - f(a) + f(a + x_2) - f(a)$.

Proof. Indeed, (i) implies (ii) since the latter is a particular case of the defining inequality for convexity. Suppose now that (ii) holds, that is, $f(i+1) - f(i) \geq f(i) - f(i-1)$, and $x = x_1 + x_2$, where x_1 and x_2 have the same sign as x . Then applying this inequality $|x_1|$ times we get $f(a+x) - f(a+x_1) \geq f(a+x_2) - f(a)$. (If $x > 0$, this follows directly; if $x < 0$ then in the same way $f(a) - f(a+x_2) \geq f(a+x_1) - f(a+x)$, which is the same.) This inequality (after rearranging) is the same as (iii). Until now we have not even used the assumption about the domain.

We finally prove that (iii) implies (i). Let $x, y, z \in D$, $z = \lambda x + (1 - \lambda)y$. Suppose without loss of generality $x = z + r$, $y = z - s$, $r, s \in \mathbb{N}$, and prove

$$(s+r)f(z) \leq sf(x) + rf(y).$$

Since by the condition all integers between $z+r$ and $z-s$ are in the domain of f , we have: $f(z+r) - f(z) = f(z+r) - f(z+r-1) + f(z+r-1) - f(z+r-2) + \dots + f(z+1) - f(z) \geq r(f(z+1) - f(z))$, and similarly $f(z) - f(z-l) \leq s(f(z+1) - f(z))$, whence $f(z) - f(z-l) \leq (s/r)(f(z+r) - f(z))$. Rearranging, we get exactly the inequality we had to prove. \square

We need (4) to hold only for improving vectors, and this property does not imply convexity. Conversely, convexity is also not sufficient for (4) to hold: define $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ with $f(x, y) := \max(x, y)$, and let $u = (0, 0)$, $\lambda := (2k+1, 2k+1)$, $\lambda_1 := (k+1, k)$, $\lambda_2 := (k, k+1)$, where $k \in \mathbb{N}$. Then

$$f(u+\lambda) - f(u) = 2k+1 < (k+1) + (k+1) = (f(u+\lambda_1) - f(u)) + (f(u+\lambda_2) - f(u)).$$

3.2 Minsquare Factors under Classical Constraints

If we want to solve the minsquare problem for constrained subgraphs, that is to determine the minimum of the sum of squares of the degrees of subgraphs satisfying some additional requirements, we do not really get significantly more difficult problems. This is at least the case if the requirements are the ‘classical’ upper, lower bound or parity constraints for a subset of vertices.

For such problems (3) can still be applied and the improving path theorems hold. We state the most general consequence concerning the complexity of the minsquare of constrained graph factors, that is, (u, l) -factors with parity constraints:

Theorem 4. *Let $G = (V, E)$ be a graph, $k \in \mathbb{N}$, $l, u : V \rightarrow \mathbb{N}$ and $T \subseteq V$. Then $F \subseteq E$, $|F| = k$ minimizing $\sum_{v \in V} d_F^2(v)$ under the constraint $l(v) \geq d_F(v) \geq u(v)$ for all $v \in V$ and such that $d_F(v)$ has the same parity as $u(v)$ for all $v \in T$, can be found in polynomial time.*

The sum of squares objective function can be replaced here by any objective function mentioned in the previous subsection. The cardinality constraint can actually be replaced by a degree constraint on an added new vertex x_0 . Again, the linear case is much easier. For instance the minimum weight k -cardinality matching problem can be solved by adding a new vertex, joining it to every $v \in V(G)$ and requiring it to be of degree $n - 2k$ and requiring every $v \in V$ to be of degree 1. In polyhedral terms this is an exercise on Schrijver’s web page [16] and in Exercise 6.9 of [4] – about the integrality of the intersection of f -factor polyhedra with the hyperplane $x_1 + \dots + x_n = k$ to which we provided thus one simple solution, and another through our main result, both different from the one suggested in [4].)

3.3 Jump Systems

A *jump system* [3] is a set $J \subseteq \mathbb{Z}^n$ with the property that for all $x, y \in J$ and each vector $x' \notin J$ at one step from x towards y there exists a vector $x'' \in J$ at one step from x' towards y . We say that x' is at one step from x towards y if $x' - x$ is ± 1 times a unit vector and the non-zero coordinate of $x' - x$ has the same sign as the corresponding coordinate of $y - x$. More generally, x' is at s steps from x towards y if $|x' - x| = s$ and $|x' - x| + |y - x'| = |y - x|$, that is, the l_1 -distance of x' from x is s , and it is on the ‘ l_1 -line’ that joins x and y .

Jump systems generalize, among others, the family of bases or independent sets of matroids, delta-matroids, and degree sequences of the subgraphs of a graph (some of these statements are easy, some others are more difficult to prove, see [3], [5]). Their convex hulls constitute the most general class of polyhedra for which the greedy algorithm works.

In the proof of Theorem 1 the key-fact is that starting from a given graph factor one can arrive at any other one by small changes in the right direction (3). This property makes sense for sets of vectors in general, but it requires a more restrictive property than the definition of jump systems:

Let us say that $L \subseteq \mathbb{Z}^V$ is a leap-system if for all $x, y \in L$, $y - x = l_1 + \dots + l_m$ ($m, \in \mathbb{N}$, $l_i \in \mathbb{Z}^V$, $i = 1, \dots, m$) where, $|y - x| = |l_1| + \dots + |l_m|$, $|l_i| \leq 2$, ($i = 1, \dots, m$) and $x + \sum_{i \in Q} l_i \in L$, for $Q \subseteq \{1, \dots, m\}$, $|Q| \leq 2$.

It is straightforward to check that Theorem 1 and the algorithms are valid without any change if degree sequences are replaced by leap-systems.

The definition of an improving vector is now the following:

Let L be a leap-system and $l \in L$. Then $\kappa \in \mathbb{Z}^V$ is an *improving vector* with respect to l if $b = l + \kappa \in L$, and $\sum_{v \in V} |\kappa(v)| \leq 4$, $f(b) < f(l)$, while $\sum_{v \in V} b(v) = \sum_{v \in V} l(v)$.

Theorem 5. *Let L be a leap system. If $l \in L$ is not optimal, then there exists an improving vector.*

As a consequence the sum of one dimensional convex functions can be optimized on leap systems intersected with hyperplanes, in polynomial time.

3.4 Weighted Minsquare, Maxsquare, Minfix, or Maxfix Cover

Let us first see what we can say about the weighted minsquare problem. Let a_1, \dots, a_n be an instance of a partition problem. Define a graph $G = (V, E)$ on $n + 2$ vertices $V = \{s, t, 1, \dots, n\}$, and join both s and t to i with an edge of weight a_i . (The degree of both s and t is n and that of all the other vertices is 2.)

Prescribe the vertices of degree 2 (that is, the vertices i , $i = 1, \dots, n$) to have exactly one incident edge in the factor, that is, the upper and lower bounds (see Section 3.2) are 1. Then the contribution of these vertices to the sum of squares of the degrees is fix and the sum of the contributions of s and t is at least $((a_1 + \dots + a_n)/2)^2$, with equality if and only if the PARTITION problem has a solution with these data. (NP-completeness may hold without degree constraints as well.)

We showed in the Introduction (Section 1) that the maxfix cover problem in the line graph of G can be reduced to the minsquare problem in G , which in turn is polynomially solvable. We also exhibited how the relation between transversals and stable sets extends to our more general problems. The following two extensions arise naturally and both turn out to be NP-hard:

In the context of maxfix covers it is natural to put weights on the hyperedges. Associate weights to the hyperedges and the total weight of hyperedges that are covered is to be maximized with a fixed number of elements. The edge-weighted maxfix cover problem is the graphic particular case of this, and even this is NP-hard, and even for cliques: the maxfix (vertex) cover problem for a graph $G = (V, E)$ is the same as the weighted maxfix cover problem for the complete graph on V with edge-weights 1 if $e \in E$, and 0 otherwise. Furthermore, a clique is a line graph (for instance of a star) so *path-weighted maxfix cover is NP-hard for stars and even for 0 – 1 weights*.

The maxsquare problem (and accordingly the minfix cover problem in line graphs) is NP-hard ! Indeed, let's reduce the problem of deciding whether a clique

of size r exists in the graph $G = (V, E)$ to a maxsquare problem in $\hat{G} = (\hat{V}, \hat{E})$ (equivalently, to a min cover problem in $L(\hat{G})$) where \hat{G} is defined as follows: subdivide every edge of G into two edges with a new vertex, and for all $v \in V$ add $\Delta_G - d_G(v)$ edges to new vertices of degree 1 each. We suppose that G does not have loops or parallel edges.

Clearly, \hat{G} is a bipartite graph, where the two classes are $A := V$ and $B := \hat{V} \setminus V$. In A all the degrees are equal to Δ , and in B they are all at most 2.

If $K \subseteq V$ is a clique of size r in G , then let $F_K \subseteq E(\hat{G})$ be defined as $F_K := \delta_{\hat{G}}(K)$ where $K \subseteq V = A \subseteq \hat{V}$. Clearly, $|F_K| = r\Delta$. The interested reader may check the following assertions: F_K is a maxsquare subgraph of cardinality $r\Delta$, and conversely, a subgraph $F \subseteq V$ of cardinality $r\Delta$ where the sum of the squares of the degrees is at least that of F_K above, satisfies $F = F_K$ for some $K \subseteq V$, $|K| = r$ that induces a complete subgraph. Therefore the problem of deciding whether G has a clique of size r can be polynomially reduced to a maxsquare problem, showing that the latter is also NP-complete.

Among all these problems the most interesting is maybe the one we could solve: indeed, it generalizes the maximum matching problem and the methods are also based on those of matching theory.

Acknowledgment: We are thankful to Maurice Queyranne and Akihisa Tamura for their questions about the relation of (4) to convexity that generated the remarks in Section 3.1.

References

1. N. Apollonio, L. Caccetta, B. Simeone, Cardinality Constrained Path Covering Problems in Trees, Tech.Rep. n. 18, Department of Statistics, Probability, and Applied Statistics, University of Rome “La Sapienza”, 2002.
2. N. Apollonio, L. Caccetta, B. Simeone, Cardinality Constrained Path Covering Problems in Grid Graphs, *Networks*, to appear.
3. A. Bouchet, W. Cunningham, Delta-matroids, Jump-systems and Bisubmodular polyhedra, SIAM J. Discrete Mathematics, February 1995.
4. W. Cook, W. Cunningham, W. Pulleyblank, A. Schrijver, Combinatorial Optimization, John Wiley & Sons, Inc, 1998.
5. W.H. Cunningham and J. Green-Krótki, b -matching degree-sequence polyhedra, *Combinatorica* **11** (1991) 219-230.
6. J.R. Edmonds, Maximum matching and a polyhedron with 0,1-vertices, J. Res. Nat. Bur. Standards Sect. B (1968), 125–130.
7. J.R. Edmonds, E.L. Johnson, Matching: a well-solved class of integer linear programs, in Combinatorial Structures and Their Applications, Calgary, Alberta, (1969), Guy, Hanani, Sauer, Schönheim eds.
8. A. M. H. Gerards, ‘Matching’, in the Handbook of Operations Research and Management Science, Volume 7, ‘Network Models’ (1995), 135–224.
9. M. Grötschel, L. Lovász, A. Schrijver, “Geometric Algorithms and Combinatorial Optimization”, Springer Verlag, Berlin Heidelberg (1988).

10. L. Lovász, On the structure of factorizable graphs, *Acta Math. Acad. Sci. Hungar.* 23, 1972, 179–195.
11. L. Lovász, The factorization of graphs II, *Acta Math. Acad. Sci. Hungar.* 23, 1972, 223–246.
12. L. Lovász, M.D. Plummer, Matching Theory, Akadémiai Kiadó, Budapest, 1986.
13. K. Murota, Discrete convex analysis, *Mathematical Programming*, **83** (1998), 313–371
14. E. Petrank, The Hardness of Approximation: Gap Location, *Computational Complexity* **4** (1994) 133–157.
15. W. R. Pulleyblank, ‘Matchings and stable sets’, in Graham, Grötschel, Lovász (eds.), *Handbook of Combinatorics* (Elsevier Science, 1995).
16. A. Schrijver, A Course in Combinatorial Optimization (lecture notes), <http://homepages.cwi.nl/~lex/>, October, 2000.

Low-Dimensional Faces of Random 0/1-Polytopes^{*}

Volker Kaibel

DFG Research Center *Mathematics for key technologies*
TU Berlin MA 6–2
Straße des 17. Juni 136
10623 Berlin
Germany
kaibel@math.tu-berlin.de

Abstract. Let P be a random 0/1-polytope in \mathbb{R}^d with $n(d)$ vertices, and denote by $\varphi_k(P)$ the k -face density of P , i.e., the quotient of the number of k -dimensional faces of P and $\binom{n(d)}{k+1}$. For each $k \geq 2$, we establish the existence of a sharp threshold for the k -face density and determine the values of the threshold numbers τ_k such that, for all $\varepsilon > 0$,

$$\mathbb{E}[\varphi_k(P)] = \begin{cases} 1 - o(1) & \text{if } n(d) \leq 2^{(\tau_k - \varepsilon)d} \text{ for all } d \\ o(1) & \text{if } n(d) \geq 2^{(\tau_k + \varepsilon)d} \text{ for all } d \end{cases}$$

holds for the expected value of $\varphi_k(P)$. The threshold for $k = 1$ has recently been determined in [1].

In particular, these results indicate that the high face densities often encountered in polyhedral combinatorics (e.g., for the cut-polytopes of complete graphs) should be considered more as a phenomenon of the general geometry of 0/1-polytopes than as a feature of the special combinatorics of the underlying problems.

1 Introduction and Results

Over the last decades, investigations of various special classes of 0/1-polytopes (convex hulls of sets of 0/1-points) have not only lead to beautiful structural results on combinatorial optimization problems, but also to powerful algorithms. Consequently, there has been some effort to learn more about the general class of 0/1-polytopes (see [2]).

In the 1980's, e.g., several results on the graphs of 0/1-polytopes have been obtained, most notably Naddef's proof [3] showing that they satisfy the Hirsch-conjecture. A quite spectacular achievement in 2000 was Bárány and Pór's theorem [4] stating that random 0/1-polytopes (within a certain range of vertex numbers) have super-exponentially (in the dimension) many facets. Their proof

* This work was done while the author was a member of the *Mathematical Sciences Research Institute* at Berkeley, CA, during Oct/Nov 2003.

is based on the methods developed in the early 1990's by Dyer, Füredi, and McDiarmid [5], in order to show that the expected volume of a random d -dimensional 0/1-polytope with n vertices drops from (almost) zero to (almost) one very quickly with n passing the threshold $2^{(1-(\log e)/2)d}$.

While Bárány and Pór's result sheds some light on the highest-dimensional faces of random 0/1-polytopes, we investigate their lower dimensional faces in this paper. For a polytope P with n vertices and some $k \in [\dim P]$ (with $[a] := \{1, 2, \dots, \lfloor a \rfloor\}$), we call

$$\varphi_k(P) := \frac{f_k(P)}{\binom{n}{k+1}}$$

the k -face density of P , where $f_k(P)$ is the number of k -dimensional faces of P . Clearly, we have $0 < \varphi_k(P) \leq 1$, and $\varphi_k(P) = 1$ holds if and only if P is $(k+1)$ -neighbourly in the usual polytope theoretical sense (see, e.g., [6]).

The 1-face density $\varphi_1(P)$ is the density of the graph of P . In this case, a threshold result for random 0/1-polytopes has recently been obtained in [1]. However, for specific classes of 0/1-polytopes, high k -face densities have been observed also for larger values of k . For example, the cut-polytopes of complete graphs have 2-face density equal to one (and thus, also 1-face density equal to one), i.e., every triple of vertices makes a triangle-face (see [7, 8]). Note that the cut-polytopes of complete graphs have $2^{\Theta(\sqrt{d})}$ vertices.

Here, we obtain that there is a sharp threshold for the k -face density of random 0/1-polytopes for all (fixed) k . The threshold values nicely extend the results for $k = 1$, while the proof becomes more involved and needs a heavier machinery (the one developed in the above mentioned paper by Dyer, Füredi, and McDiarmid). As a pay-back, the proof, however, reveals several interesting insights into the geometry of (random) 0/1-polytopes.

1.1 Results

Let us fix some $k \in \{1, 2, \dots\}$, set $r := k + 1$, and let $n : \mathbb{N} \rightarrow \mathbb{N}$ be a function (with $n(d) \in [2^d]$).

Define

$$V_d := \{0, 1\}^d \quad \text{and} \quad Q_d := [0, 1]^d = \text{conv } V_d ,$$

and consider the following two models of random 0/1-polytopes.

For the first one, choose W uniformly at random from the $n(d)$ -element subsets of V_d , and define $P_1 := \text{conv } W$. This is the model referred to in the abstract.

For the second one, choose $S_1, \dots, S_r, X_1, \dots, X_{n(d)-r} \in V_d$ independently uniformly at random, and define

$$S := \{S_1, \dots, S_r\} , \quad X := \{X_1, \dots, X_{n(d)-r}\} , \quad P_2 := \text{conv } (X \cup S) .$$

The main part of the paper will be concerned with the proof of a threshold result (Theorem 1) within the second model. If, for some $\varepsilon > 0$, $n(d) \leq 2^{(\frac{1}{2}-\varepsilon)d}$

holds for all d , then $S_1, \dots, S_r, X_1, \dots, X_{n(d)-r}$ are pairwise different with high probability:

$$\mathbb{P} [|S \cup X| = n(d)] = 1 - o(1) \quad (1)$$

This will allow us to deduce from Theorem 1 the threshold result within the first model promised in the abstract.

Throughout the paper, $\log(\cdot)$ and $\ln(\cdot)$ will denote the binary and the natural logarithm, respectively. For $0 < \xi < 1$, define

$$h(\xi) := \xi \log \frac{1}{\xi} + (1 - \xi) \log \frac{1}{1 - \xi}$$

(i.e., $h(\cdot)$ is the binary entropy function). Let us define

$$H_r := \frac{1}{2^r - 2} \sum_{i \in [r-1]} \binom{r}{i} h\left(\frac{i}{r}\right)$$

and

$$\tilde{\tau}_r = 1 - (1 - 2^{1-r})H_r .$$

Note that we have $H_2 = 1$ and $0 < H_r < 1$ for $r \geq 3$.

Theorem 1. *Let $r \in \{3, 4, \dots\}$ and $\varepsilon > 0$.*

1. *If $n(d) \leq 2^{(\tilde{\tau}_r - \varepsilon)d}$ holds for all d , then we have*

$$\mathbb{P} [\text{conv } S \text{ is a face of } P_2] = 1 - o(1) .$$

2. *If $n(d) \geq 2^{(\tilde{\tau}_r + \varepsilon)d}$ holds for all d , then we have*

$$\mathbb{P} [P_2 \cap \text{aff } S \text{ is a face of } P_2] = o(1) .$$

From the evolution result on the density of the graphs of random 0/1-polytopes obtained in [1] one readily derives that the statement of Theorem 1 is also true for $r = 2$ (note $\tilde{\tau}_2 = \frac{1}{2}$).

Using Theorem 1 (for $r \in \{2, 3, \dots\}$), we can now prove the main result of the paper, where for $k \in \{1, 2, \dots\}$ we denote

$$\tau_k := \tilde{\tau}_{k+1} = 1 - (1 - 2^{-k})H_{k+1} .$$

Theorem 2. *Let $k \in \{1, 2, \dots\}$, $\varepsilon > 0$, and $n : \mathbb{N} \rightarrow \mathbb{N}$ be any function. For each $d \in \mathbb{N}$, choose an $n(d)$ -element subset W of $\{0, 1\}^d$ uniformly at random, and set $P := \text{conv } W$. Then*

$$\mathbb{E} [\varphi_k(P)] = \begin{cases} 1 - o(1) & \text{if } n(d) \leq 2^{(\tau_k - \varepsilon)d} \text{ for all } d \\ o(1) & \text{if } n(d) \geq 2^{(\tau_k + \varepsilon)d} \text{ for all } d \end{cases}$$

holds for the expected k -face density of P .

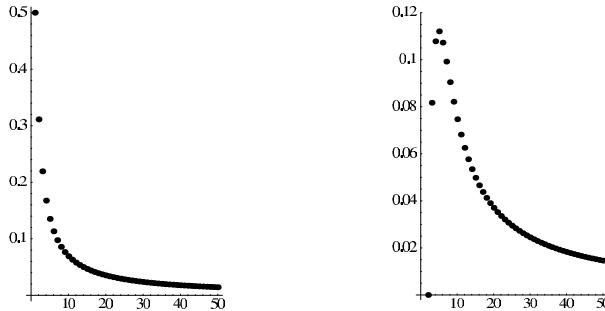


Fig. 1. The values τ_k for $k \geq 1$ and $1 - H_r$ (see Proposition 1) for $r \geq 2$.

Proof. Let us first consider the case $n(d) \leq 2^{(\tau_k - \varepsilon)d}$. We adopt the notation introduced in order to describe the first random model; in particular, $P_1 = P = \text{conv } W$. Since $r = k + 1$ is constant, S (from the second random model) will consist of $k+1$ affinely independent points with (very) high probability for large d (see [9]). Thus, the first part of Theorem 1 here implies

$$\mathbb{P}[\text{conv } S \text{ is a } k\text{-dimensional face of } P_2] = 1 - o(1).$$

Due to (1) (note $\tau_k \leq \frac{1}{2}$), this yields

$$\mathbb{P}[\text{conv } T \text{ is a } k\text{-dimensional face of } P_1] = 1 - o(1)$$

for T chosen uniformly at random from the $(k+1)$ -subsets of (the random $n(d)$ -set) W . But this probability obviously is a lower bound for $\mathbb{E}[\varphi_k(P_1)]$, which proves the first part of the theorem.

Now, we consider the case $n(d) \geq 2^{(\tau_k + \varepsilon)d}$. Similarly to the first case, the second part of Theorem 1 here implies

$$\mathbb{P}[P_2 \cap \text{aff } S \text{ is a face of } P_2 \mid |S| = k+1] = o(1). \quad (2)$$

Furthermore, it is easy to see that

$$\begin{aligned} \mathbb{P}[P_2 \cap \text{aff } S \text{ is a face of } P_2 \mid |S \cup X| = n(d)] \\ \leq \mathbb{P}[P_2 \cap \text{aff } S \text{ is a face of } P_2 \mid |S| = k+1] \end{aligned} \quad (3)$$

holds. From (2) and (3) one readily deduces

$$\mathbb{P}[P_1 \cap \text{aff } T \text{ is a face of } P_1] = o(1)$$

for T again chosen uniformly at random from the $(k+1)$ -subsets of W . Since the number of k -faces of a polytope is at most the number of $(k+1)$ -subsets of its vertex set for which the intersections of their affine hulls and the polytope are faces of the polytope, the latter probability is an upper bound for $\mathbb{E}[\varphi_k(P_1)]$. This proves the second part of the theorem.

1.2 Overview of the Proof of Theorem 1

The structure of the proof is as follows: First, we will (in Section 2) reduce the proof of Theorem 1 to a statement (Proposition 1) about the event that S is not contained in a proper face of the cube, i.e., S is *spanning*. (A *proper* face of a polytope is any face that is not the entire polytope, which is considered a face of itself here.) This statement finally is proved in Section 5. There we need the results of Section 3 (for treating the cases *behind* the threshold) and Section 4 (for the cases *below* the threshold).

We will use only basic facts from polytope theory (such as in the proof of Theorem 2). Consult [6] in case of doubts – or for background information.

Throughout the paper, $r \in \{3, 4, \dots\}$ will be a constant.

Acknowledgments

I am grateful to the Mathematical Sciences Research Institute at Berkeley for the generous support and the excellent conditions I enjoyed during my visit in October/November 2003, when this work was done. I thank Günter M. Ziegler for comments on an earlier version of the paper.

2 Reduction to the Spanning Case

From now on, we stick to the second model of randomness. Thus, for some function $n : \mathbb{N} \rightarrow \mathbb{N}$, we choose the points $S_1, \dots, S_r, X_1, \dots, X_{n(d)-r} \in V_d$ independently uniformly at random, and let $S := \{S_1, \dots, S_r\}$, $X := \{X_1, \dots, X_{n(d)-r}\}$, and $P := \text{conv}(X \cup S)$. Denote by $F(S)$ the smallest face of the cube Q_d that contains S . Clearly, $P \cap F(S)$ is a face of P . Let $d(S)$ be the dimension of $F(S)$ (i.e., $d(S)$ is the number of coordinates where not all elements of S agree). If $F(S) = Q_d$ (i.e., $d(S) = d$), then we call S *spanning*.

In Section 5, we will prove the following result (where ∂ denotes the boundary operator).

Proposition 1. *Let $r \in \{3, 4, \dots\}$ and $\varepsilon > 0$.*

1. *If $n(d) \leq 2^{(1-H_r-\varepsilon)d}$ holds for all d , then we have*

$$\mathbb{P}[\text{conv } S \text{ is a face of } P \mid S \text{ is spanning}] = 1 - o(1).$$

2. *If $n(d) \geq 2^{(1-H_r+\varepsilon)d}$ holds for all d , then we have*

$$\mathbb{P}[\text{conv } S \subseteq \partial P \mid S \text{ is spanning}] = o(1).$$

Figure 1 illustrates the threshold values $1 - H_r$. The aim of the current section is to show that Proposition 1 implies Theorem 1.

2.1 Preliminaries

Let A be the $r \times d$ matrix whose rows are S_1, \dots, S_r . Clearly, $d(S)$ equals the number of columns of A which are neither \emptyset (the all-zero vector) nor $\mathbf{1}$ (the all-one vector).

The random matrix A is distributed in the same way as an $r \times d$ matrix is distributed whose columns are chosen independently uniformly at random from $\{0, 1\}^r$. For $t \in \{0, 1\}^r$ chosen uniformly at random, we have $\mathbb{P}[t \notin \{\emptyset, \mathbf{1}\}] = 1 - 2^{1-r}$.

The de Moivre-Laplace Theorem (see, e.g., [10, Chap. 7]) yields that, for every $\delta > 0$, there is a $B_\delta > 0$ such that

$$\mathbb{P} [|d(S) - (1 - 2^{1-r})d| \leq B_\delta \sqrt{d}] \geq 1 - \delta \quad (4)$$

holds for all large enough d .

For each $\delta > 0$, define

$$J_\delta(d) := \{j \in [d] : |j - (1 - 2^{1-r})d| \leq B_\delta \sqrt{d}\}.$$

Thus, by (4) we have

$$\mathbb{P}[d(S) \in J_\delta] \geq 1 - \delta \quad (5)$$

for all large enough d .

Let us denote

$$n(S) := |\{i \in [n] : X_i \in F(S)\}|.$$

2.2 The Case $n(d) \leq 2^{(\tilde{\tau}_r - \varepsilon)d}$

From elementary polytope theory one derives

$$\text{conv } S \text{ is a face of } P \Leftrightarrow \text{conv } S \text{ is a face of } P \cap F(S). \quad (6)$$

Let $\delta > 0$ be fixed and let $j_{\min} \in J_\delta$ such that

$$\begin{aligned} \mathbb{P}[\text{conv } S \text{ is a face of } P \mid d(S) = j_{\min}] \\ = \min \{\mathbb{P}[\text{conv } S \text{ is a face of } P \mid d(S) = j] : j \in J_\delta(d)\}. \end{aligned}$$

Then we have

$$\left| d - \frac{j_{\min}}{1 - 2^{1-r}} \right| = o(j_{\min}).$$

We therefore obtain

$$\begin{aligned} \mathbb{E} [n(S) \mid d(S) = j_{\min}] &= 2^{j_{\min}-d}(n(d) - r) \\ &\leq 2^{j_{\min}-d+(\tilde{\tau}_r-\varepsilon)d+o(j_{\min})} \\ &\leq 2^{\frac{1-2^{1-r}+\tilde{\tau}_r-1-\varepsilon}{1-2^{1-r}}j_{\min}+o(j_{\min})}. \end{aligned} \quad (7)$$

The fraction in the exponent equals $1 - H_r - \varepsilon'$ where $\varepsilon' := \frac{\varepsilon}{1-2^{1-r}} > 0$. By Markov's inequality, we obtain

$$\mathbb{P}[n(S) \leq 2^{(1-H_r-\varepsilon'/2)j_{\min}} | d(S) = j_{\min}] = 1 - o(1). \quad (8)$$

Proposition 1 implies

$$\begin{aligned} \mathbb{P}[\text{conv } S \text{ is a face of } P \cap F(S) | d(S) = j_{\min}, n(S) \leq 2^{(1-H_r-\varepsilon'/2)j_{\min}}] \\ = 1 - o(1). \end{aligned}$$

Together with (8), the definition of j_{\min} , and (5), this implies

$$\mathbb{P}[\text{conv } S \text{ is a face of } P \cap F(S)] = 1 - o(1),$$

which, by (6), proves the first part of Theorem 1.

2.3 The Case $n(d) \geq 2^{(\tilde{\tau}_r+\varepsilon)d}$

Again, elementary polytope theory tells us

$P \cap \text{aff } S$ is a face of P

$$\Rightarrow P \cap \text{aff } S = P \cap F(S) \text{ or } \text{conv } S \subseteq \partial(P \cap F(S)). \quad (9)$$

We omit the calculations that are necessary to prove the following lemma.

Lemma 1. *Let $\alpha, \beta, \gamma > 0$ with $\alpha + \beta > 1 + \beta\gamma$, $\tilde{n}(d) := \lfloor 2^{\alpha d} \rfloor$, $j(d) = \beta d + o(d)$, and let F be any $j(d)$ -dimensional face of Q_d . If $X_1, \dots, X_{\tilde{n}(d)}$ are chosen independently uniformly at random from V_d , then we have*

$$\mathbb{P}[\left|\{i \in [\tilde{n}(d)] : X_i \in F\}\right| \geq 2^{\gamma j(d)}] = 1 - o(1).$$

Now we can prove the second part of Theorem 1 (using Proposition 1). Let $\delta > 0$ be fixed and let $j_{\max} \in J_\delta$ such that

$$\begin{aligned} \mathbb{P}[\text{conv } S \subseteq \partial(P \cap F(S)) | d(S) = j_{\max}] \\ = \max \{ \mathbb{P}[\text{conv } S \subseteq \partial(P \cap F(S)) | d(S) = j] : j \in J_\delta(d) \}. \end{aligned}$$

With $\alpha := \tilde{\tau}_r + \varepsilon$, $\beta := 1 - 2^{1-r}$, and $\gamma := 1 - H_r + \varepsilon$, one easily verifies $\alpha + \beta > 1 + \beta\gamma$. Since $j_{\max} = (1 - 2^{1-r})d + o(d)$ we thus obtain from Lemma 1

$$\mathbb{P}[n(S) \geq 2^{(1-H_r+\varepsilon)j_{\max}} | d(S) = j_{\max}] = 1 - o(1). \quad (10)$$

The second part of Proposition 1 implies

$$\mathbb{P}[\text{conv } S \subseteq \partial(P \cap F(S)) | d(S) = j_{\max}, n(S) \geq 2^{(1-H_r+\varepsilon)j_{\max}}] = o(1).$$

Furthermore, since $\dim(\text{aff } S)$ is constant, we obviously have

$$\mathbb{P}[P \cap \text{aff } S = P \cap F(S) | d(S) = j_{\max}, n(S) \geq 2^{(1-H_r+\varepsilon)j_{\max}}] = o(1).$$

Together with (10), the definition of j_{\max} , and (5), the latter two equations even hold for the corresponding unconditioned probabilities. Thus, we have

$$\mathbb{P}[\text{conv } S \subseteq \partial(P \cap F(S)) \text{ or } P \cap \text{aff } S = P \cap F(S)] = o(1),$$

which, due to (9), proves the second part of Theorem 1.

3 Membership Probabilities

Here, we derive (from Dyer, Füredi, and McDiarmid's paper [5]) suitable lower bounds on $n(d)$ that, for specified points of Q_d , guarantee their membership in our random 0/1-polytopes with high probability.

For any $z \in Q_d$, let us define

$$p(z) := \frac{1}{2^d} \min \left\{ |U \cap V_d| : U \subset \mathbb{R}^d \text{ (closed affine) halfspace, } z \in U \right\}.$$

For each $\alpha > 0$, denote

$$Q_d^\alpha := \{z \in Q_d : p(z) \geq 2^{-\alpha d}\}.$$

For $z = (\zeta_1, \dots, \zeta_d) \in \text{int } Q_d$ (the interior of Q_d), define

$$H(z) := \frac{1}{d} \sum_{j \in [d]} h(\zeta_j).$$

From Lemmas 2.1 and 4.1 of [5] one can deduce the following fact. Let us mention that in particular the proof of Lemma 4.1 (needed for part (2) of Lemma 2) is quite hard. It is the core of Dyer, Füredi, and McDiarmid's beautiful paper.

Lemma 2. *Let $\alpha, \varepsilon > 0$.*

1. *If $\tilde{n}(d) \geq 2^{(\alpha+\varepsilon)d}$ holds for all d , and $X_1, \dots, X_{\tilde{n}(d)} \in V_d$ are chosen independently uniformly at random, then we have*

$$\mathbb{P}[Q_d^\alpha \subseteq \text{conv}\{X_1, \dots, X_{\tilde{n}(d)}\}] = 1 - o(1).$$

2. *For large enough d ,*

$$\{z \in \text{int } Q_d : H(z) \geq 1 - \alpha + \varepsilon\} \subseteq Q_d^\alpha$$

holds.

The following straight consequence (choose $\alpha := 1 - \beta + \varepsilon/2$) of Lemma 2 is the key to the proof of the second part of Proposition 1.

Corollary 1. *If $\beta > 0$, $\tilde{n}(d) \geq 2^{(1-\beta+\varepsilon)d}$ for all d , and $X_1, \dots, X_{\tilde{n}(d)} \in V_d$ are chosen independently uniformly at random, then we have*

$$\mathbb{P}[\{z \in \text{int } Q_d : H(z) \geq \beta\} \subseteq \text{conv}\{X_1, \dots, X_{\tilde{n}(d)}\}] = 1 - o(1).$$

4 Shallow Cuts of the Cube

This section is the heart of the proof of (the first part of) Proposition 1.

For $m \in \{1, 2, \dots\}$, let $A(m)$ be an $r \times M$ matrix with $M := (2^r - 2)m$ that has as its columns m copies of each vector $v \in \{0, 1\}^r \setminus \{\emptyset, \mathbf{1}\}$. This choice is motivated by the following fact (which is, however, irrelevant in this section): If S_1, \dots, S_r are chosen independently uniformly at random from V_M , then the multiplicity m of each vector $v \in \{0, 1\}^r$ among the columns of $A(m)$ equals the expected number of appearances of v as a column of the matrix with rows S_1, \dots, S_r — conditioned on the event that S is spanning.

Let $s_1, \dots, s_r \in \{0, 1\}^M$ be the rows of $A(m)$, and let, for $1 \leq i \leq r-1$, $L(i)$ be the set of indices of columns that have precisely i ones. We have $|L(i)| = \binom{r}{i}m$. Denote by $\sigma(i)$ the number of ones that any of the rows has in columns indexed by $L(i)$ (these numbers are equal for all rows). Obviously, we have $\sigma(i) = \frac{i}{r}\binom{r}{i}m$.

Let $b := (\beta_1, \dots, \beta_M)$ be the barycenter of the rows s_1, \dots, s_r . For each $j \in [M]$ we thus have $\beta_j = \frac{i(j)}{r}$, if $j \in L(i(j))$. Consequently (with the definition of $H(\cdot)$ from Section 3),

$$H(b) = \frac{1}{M} \sum_{i \in [r-1]} \binom{r}{i} m h\left(\frac{i}{r}\right) = H_r. \quad (11)$$

From Section 3 (see Lemma 2) we know that no hyperplane in \mathbb{R}^M that contains b can therefore cut off significantly *less* than $2^{H_r M}$ points from V_M , and that there are indeed hyperplanes containing b that do also not cut off significantly *more* than $2^{H_r M}$ cube vertices. However, for our purposes, it will be necessary to know that there is a hyperplane containing not only b , but even the entire set $\{s_1, \dots, s_r\}$, and nevertheless cutting off not significantly more than $2^{H_r M}$ cube vertices.

The next result guarantees the existence of such a hyperplane, i.e., a certain shallow cut of the cube. Its proof will also reveal the basic reason for the appearance of the entropy function $h(\cdot)$: It is due to the well-known fact that, for any constant $\alpha > 0$,

$$\sum_{p \in [\alpha q]} \binom{q}{p} = 2^{h(\alpha)q + o(q)} \quad (12)$$

(see, e.g., [11, Chap. 9, Ex. 42]).

Proposition 2. *There are coefficients $\alpha_1, \dots, \alpha_{r-1} \in \mathbb{R}$, such that the inequality*

$$\sum_{i \in [r-1]} \sum_{j \in L(i)} \alpha_i \xi_j \leq \sum_{i \in [r-1]} \alpha_i \sigma(i) \quad (13)$$

has at most $2^{H_r M + o(M)}$ 0/1-solutions $(\xi_1, \dots, \xi_M) \in \{0, 1\}^M$. (By construction, the 0/1-points s_1, \dots, s_r satisfy (13) with equality.)

Proof. Throughout the proof, we denote the components of any vectors $a, l, z \in \mathbb{R}^{r-1}$ by α_i , λ_i , and ζ_i , respectively.

For every $a \in \mathbb{R}^{r-1}$ and $l \in \mathbb{N}^{r-1}$, denote by $\omega_a(l)$ the number of 0/1-solutions to (13) with precisely λ_i ones in components indexed by $L(i)$ and define

$$\omega(l) := \prod_{i \in [r-1]} \binom{\binom{r}{i}m}{\lambda_i}.$$

With

$$L_a := \left\{ l \in \mathbb{N}^{r-1} : \sum_{i \in [r-1]} \alpha_i \lambda_i \leq \sum_{i \in [r-1]} \alpha_i \sigma(i) \right\}$$

we thus have

$$\omega_a(l) = \begin{cases} \omega(l) & \text{if } l \in L_a \\ 0 & \text{otherwise} \end{cases}.$$

Consequently, the number of 0/1-points satisfying (13) is precisely

$$\sum_{l \in L_a} \omega(l). \quad (14)$$

If, for some i , we have $\lambda_i > \binom{r}{i}m$, then clearly $\omega(l) = 0$. Thus, the number of nonzero summands in (14) is $O(m^r)$. Below, we will exhibit a vector $a \in \mathbb{R}^{r-1}$ of (constant) coefficients that satisfies, with $z^* := (\sigma(1), \dots, \sigma(r-1))$,

$$\omega(l) \leq \omega(z^*) 2^{o(M)} \quad (15)$$

for all $l \in L_a$. This will eventually prove the proposition, since we have

$$\begin{aligned} \omega(z^*) &= \prod_{i \in [r-1]} \binom{\binom{r}{i}m}{\sigma(i)} \\ &= \prod_{i \in [r-1]} \binom{\binom{r}{i}m}{(i/r)\binom{r}{i}m} \\ &= \prod_{i \in [r-1]} 2^{h(i/r)\binom{r}{i}m + o(m)} \\ &= 2^{\sum_{i \in [r-1]} h(i/r)\binom{r}{i}m + o(m)} \\ &= 2^{H_r M + o(M)} \end{aligned}$$

(where the third equation is due to (12), and for the the last one, see (11)).

We now approximate the function $\omega(\cdot)$ by Sterling's formula (see, e.g., [11, Eq. (9.40)])

$$N! = \Theta\left(\sqrt{N} \frac{N^N}{e^N}\right).$$

For simplicity, we define $M_i := \binom{r}{i}m$. Thus we obtain

$$\omega(l) \leq O(M^r) \prod_{i \in [r-1]} \frac{M_i^{M_i}}{\lambda_i^{\lambda_i} (M_i - \lambda_i)^{M_i - \lambda_i}}$$

(with $0^0 = 1$). Let us define the closed box

$$B := [0, M_1] \times [0, M_2] \times \cdots \times [0, M_{r-1}] ,$$

the map $\eta : B \rightarrow \mathbb{R}$ via

$$\eta(z) := \prod_{i \in [r-1]} \frac{M_i^{M_i}}{\zeta_i^{\zeta_i} (M_i - \zeta_i)^{M_i - \zeta_i}} ,$$

and the halfspace

$$U_a := \left\{ z \in \mathbb{R}^{r-1} : \sum_{i \in [r-1]} \alpha_i \zeta_i \leq \sum_{i \in [r-1]} \alpha_i \sigma(i) \right\} .$$

We have

$$\{l \in L_a : \omega(l) > 0\} \subseteq B \cap U_a .$$

By the continuity of η on B it hence suffices to determine $a \in \mathbb{R}^{r-1}$ such that $\eta(z^*) \geq \eta(z)$ holds for all $z \in U_a \cap \text{int } B$. Note that z^* itself is contained in the interior $\text{int } B$ of the box B , where η is a differentiable function.

In fact, since $\ln(\cdot)$ is monotonically increasing, we may equivalently investigate the function $\tilde{\eta} : \text{int } B \rightarrow \mathbb{R}$ defined via

$$\tilde{\eta}(z) := \ln \eta(z) = \sum_{i \in [r-1]} M_i \ln M_i - \sum_{i \in [r-1]} (\zeta_i \ln \zeta_i + (M_i - \zeta_i) \ln(M_i - \zeta_i)) ,$$

and thus find a vector $a \in \mathbb{R}^{r-1}$ of coefficients with

$$\tilde{\eta}(z^*) \geq \tilde{\eta}(z) \quad \text{for all } z \in U_a \cap \text{int } B . \quad (16)$$

Now we choose the vector $a \in \mathbb{R}^{r-1}$ to be the gradient of $\tilde{\eta}$ at z^* . One easily calculates

$$\alpha_i = \ln \frac{M_i - \sigma(i)}{\sigma(i)} .$$

In order to prove that, with this choice, (16) holds, let $z \in U_a \cap \text{int } B$ be arbitrary ($z \neq z^*$). Define $v := z - z^*$, and consider the function $\tilde{\eta}_{z^*, z} : [0, 1] \rightarrow \mathbb{R}$ defined via $\tilde{\eta}_{z^*, z}(t) := \tilde{\eta}(z^* + tv)$. The derivative of this function on $(0, 1)$ is

$$\tilde{\eta}'_{z^*, z}(t) = \sum_{i \in [r-1]} v_i \ln \frac{M_i - \sigma(i) - tv_i}{\sigma(i) + tv_i} . \quad (17)$$

Consider any $i \in [r-1]$, and define $\varrho(t) := \frac{M_i - \sigma(i) - tv_i}{\sigma(i) + tv_i}$. If $v_i \geq 0$, then $\varrho(t) \leq \varrho(0)$, therefore, $v_i \ln \varrho(t) \leq v_i \ln \varrho(0) = \alpha_i v_i$. If $v_i < 0$, then $\varrho(t) > \varrho(0)$, and thus, $v_i \ln \varrho(t) < v_i \ln \varrho(0) = \alpha_i v_i$. Hence, in any case the i -th summand in (17) is at most as large as $\alpha_i v_i$. Therefore, we obtain

$$\tilde{\eta}'_{z^*, z}(t) \leq \sum_{i \in [r-1]} \alpha_i v_i .$$

Since $z \in U_a$, we have $\sum_{i \in [r-1]} \alpha_i v_i \leq 0$. Thus, $\tilde{\eta}'_{z^*, z}(t) \leq 0$ for all $t \in (0, 1)$. Since $\tilde{\eta}_{z^*, z}$ is continuous on $[0, 1]$, we hence conclude $\tilde{\eta}(z^*) \geq \tilde{\eta}(z)$.

5 The Spanning Case

Using the material collected in Sections 3 and 4, we will now prove Proposition 1 (and thus, as shown in Section 2) Theorem 1.

Towards this end, let $S_1, \dots, S_r, X_1, \dots, X_{n(d)-r} \in V_d$ be chosen according to the probability distribution induced by our usual distribution (choosing all points independently uniformly at random) on the event that $S := \{S_1, \dots, S_r\}$ is spanning. As before, define $S := \{S_1, \dots, S_r\}$, $X := \{X_1, \dots, X_{n(d)-r}\}$, and $P := \text{conv}(S \cup X)$.

Let A be the $r \times d$ matrix with rows S_1, \dots, S_r . Then A is a random matrix that has the same distribution as the $r \times d$ random matrix A' which arises from choosing each column independently uniformly at random from $\{0, 1\}^r \setminus \{\emptyset, \mathbb{1}\}$. Therefore, if we denote the columns of A by $t_1, \dots, t_d \in \{0, 1\}^r$, then the t_j are (independently) distributed according to the distribution

$$\mathbb{P}[t_j = t] = \frac{1}{2^r - 2} =: \pi$$

for each $t \in \{0, 1\}^r \setminus \{\emptyset, \mathbb{1}\}$.

Define

$$T_r := \{0, 1\}^d \setminus \{\emptyset, \mathbb{1}\},$$

and denote, for every $t \in T_r$,

$$J(t) := \{j \in [d] : t_j = t\}.$$

Let $m \in \mathbb{N}$ be the largest number such that $m \leq |J(t)|$ holds for all $t \in T_r$. For each t , choose an arbitrary subset $\tilde{J}(t) \subseteq J(t)$ with $|\tilde{J}(t)| = m$.

Denote by

$$\Delta_{\max} := \max \{ |J(t)| - \pi d : t \in T_r \}$$

the maximal deviation of any $|J(t)|$ from its expected value πd .

From the de Moivre-Laplace Theorem (see, e.g., [10, Chap. 7]) one deduces the following for each $t \in T_r$: For every $\gamma' > 0$ there is a $C'_{\gamma'} > 0$ such that

$$\mathbb{P}[|J(t)| - \pi d \leq C'_{\gamma'} \sqrt{d}] \geq 1 - \gamma'$$

holds for all large enough d . Since $|T_r|$ is a constant, one can even derive the following stronger result from this: For every $\gamma > 0$ there is a constant $C_\gamma > 0$ such that

$$\mathbb{P}[\Delta_{\max} \leq C_\gamma \sqrt{d}] \geq 1 - \gamma \tag{18}$$

holds for all large enough d .

Let us define

$$\tilde{D} := \bigcup_{t \in T_r} \tilde{J}(t)$$

and $\tilde{d} := |\tilde{D}| = m(2^r - 2)$. In case of $\Delta_{\max} \leq C_\gamma \sqrt{d}$, we can deduce

$$\tilde{d} \geq d - o(d). \tag{19}$$

5.1 The Case $n(d) \leq 2^{(1-H_r-\varepsilon)d}$

Let $\tilde{S}_1, \dots, \tilde{S}_r$ be the canonical projections of S_1, \dots, S_r , respectively, to the coordinates in \tilde{D} . Then $\tilde{S}_1, \dots, \tilde{S}_r$ form a matrix $A(m)$ as defined in Section 4. Denote, for each $i \in [r-1]$,

$$\tilde{L}(i) := \bigcup_{t \in T_r : \mathbf{1}^T t = i} \tilde{J}(t).$$

Due to Proposition 2, there are coefficients $\tilde{a}_1, \dots, \tilde{a}_{r-1} \in \mathbb{R}$ such that the inequality

$$\sum_{i \in [r-1]} \tilde{a}_i \sum_{j \in \tilde{L}(i)} \xi_j \leq \sum_{i \in [r-1]} \tilde{a}_i \frac{i}{r} \binom{r}{i} m =: a_0 \quad (20)$$

has at most $2^{H_r \tilde{d} + o(\tilde{d})}$ many 0/1-solutions (and $\tilde{S}_1, \dots, \tilde{S}_r$ satisfy the inequality with equality).

For each $j \in [d]$ let

$$a_j := \begin{cases} \tilde{a}_i & \text{if } j \in \tilde{L}(i) \\ 0 & \text{if } j \in [d] \setminus \tilde{D} \end{cases},$$

i.e., a_1, \dots, a_d are the coefficients of (20) considered as an inequality for \mathbb{R}^d .

The inequality

$$\sum_{j \in [d]} a_j \xi_j \leq a_0 \quad (21)$$

is satisfied with equality by S_1, \dots, S_r .

Let us, for the moment, restrict our attention to the event $\Delta_{\max} \leq C_\gamma \sqrt{d}$. Then (21) has at most

$$2^{H_r \tilde{d} + o(\tilde{d})} 2^{d-\tilde{d}} = 2^{H_r d + o(d)}$$

solutions (due to (19)).

Define the halfspace

$$U := \{(\xi_1, \dots, \xi_d) \in \mathbb{R}^d : \sum_{j \in [d]} a_j \xi_j \leq a_0\},$$

and let ∂U be its bounding hyperplane. Thus, we have

$$S_1, \dots, S_r \in \partial U \quad \text{and} \quad |U \cap V_d| \leq 2^{H_r d + o(d)}. \quad (22)$$

Since $n(d) \leq 2^{(1-H_r-\varepsilon)d}$, the expected number of points from X lying in U is

$$\frac{2^{H_r d + o(d)}}{2^d} (n(d) - r) \leq 2^{-\varepsilon d + o(d)}.$$

Therefore, by Markov's inequality,

$$\mathbb{P}[X \cap U = \emptyset \mid \Delta_{\max} \leq C_\gamma \sqrt{d}] = o(1) \quad (23)$$

From (23) and (18) we derive

$$\mathbb{P}[\partial U \cap P = \text{conv } S, X \cap U = \emptyset] = 1 - o(1),$$

which proves the first part of Proposition 1.

5.2 The Case $n(d) \geq 2^{(1-H_r+\varepsilon)d}$

From the remarks in the introduction, we know

$$\mathbb{P}[|S| = r] = 1 - o(1). \quad (24)$$

Let $\gamma > 0$ be fixed, and assume $|S| = r$, i.e., the points S_1, \dots, S_r are pairwise disjoint. Denote by $b(S) = (\beta_1, \dots, \beta_d)$ the barycenter of S . For each $t \in T_r$ and $j \in \tilde{J}(t)$, we have

$$\beta_j = \frac{\mathbf{1}^T t}{r}.$$

If $\Delta_{\max} \leq C_\gamma \sqrt{d}$ holds, we thus have (where the last equation is due to (19))

$$\begin{aligned} H(b(S)) &= \frac{1}{d} \left(\sum_{t \in T_r} m h\left(\frac{\mathbf{1}^T t}{r}\right) + o(d) \right) \\ &= \frac{1}{d} \left(\sum_{i \in [r-1]} m \binom{r}{i} h(i/r) + o(d) \right) \\ &= \frac{m(2^r - 2)}{d} H_r + o(1) \\ &= (1 - o(1)) H_r + o(1). \end{aligned}$$

Hence, in this case

$$H(b(S)) \geq H_r - \frac{\varepsilon}{2}$$

holds for large enough d . Since H is continuous, there is a neighborhood N of $b(S)$ such that $H(x) \geq H_r - \varepsilon$ holds for all $x \in N$. Due to $n(d) \geq 2^{(1-H_r+\varepsilon)d}$, Corollary 1 implies

$$\mathbb{P}[N \subseteq \text{conv } X \mid |S| = r, \Delta_{\max} \leq C_\gamma \sqrt{d}] \geq 1 - o(1).$$

Together with (24) and (18), this shows

$$\mathbb{P}[b(S) \in \text{int } P] = 1 - o(1),$$

which proves the second part of Proposition 1.

References

- [1] Kaibel, V., Remshagen, A.: On the graph-density of random 0/1-polytopes. In Arora, S., Jansen, K., Rolim, J., Sahai, A., eds.: Approximation, Randomization, and Combinatorial Optimization (Proc. RANDOM03). Volume 2764 of Lecture Notes in Computer Science., Springer (2003) 318–328

- [2] Ziegler, G.M.: Lectures on 0/1-polytopes. In: Polytopes—Combinatorics and Computation (Oberwolfach, 1997). Volume 29 of DMV Sem. Birkhäuser, Basel (2000) 1–41
- [3] Naddef, D.: The Hirsch conjecture is true for $(0, 1)$ -polytopes. Math. Programming **45** (1989) 109–110
- [4] Bárány, I., Pór, A.: On 0-1 polytopes with many facets. Adv. Math. **161** (2001) 209–228
- [5] Dyer, M.E., Füredi, Z., McDiarmid, C.: Volumes spanned by random points in the hypercube. Random Structures Algorithms **3** (1992) 91–106
- [6] Ziegler, G.M.: Lectures on Polytopes. Volume 152 of Graduate Texts in Mathematics. Springer-Verlag, New York (1995) 2nd edition: 1998.
- [7] Barahona, F., Mahjoub, A.R.: On the cut polytope. Math. Programming **36** (1986) 157–173
- [8] Déza, M.M., Laurent, M.: Geometry of Cuts and Metrics. Volume 15 of Algorithms and Combinatorics. Springer-Verlag, Berlin (1997)
- [9] Kahn, J., Komlós, J., Szemerédi, E.: On the probability that a random ± 1 -matrix is singular. J. Amer. Math. Soc. **8** (1995) 223–240
- [10] Feller, W.: An introduction to probability theory and its applications. Vol. I. Third edition. John Wiley & Sons Inc., New York (1968)
- [11] Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete mathematics. Second edn. Addison-Wesley Publishing Company, Reading, MA (1994)

On Polyhedra Related to Even Factors^{*}

Tamás Király^{1,2} and Márton Makai^{1,3}

¹ Department of Operations Research, Eötvös University,
Pázmány Péter sétány 1/C, Budapest, Hungary, H-1117.
{tkiraly, marci}@cs.elte.hu

² MTA-ELTE Egerváry Research Group,
Pázmány Péter sétány 1/C, Budapest, Hungary, H-1117.

³ Communication Networks Laboratory,
Pázmány Péter sétány 1/A, Budapest, Hungary, H-1117.

Abstract. As a common generalization of matchings and matroid intersection, W.H. Cunningham and J.F. Geelen introduced the notion of path-matching, which they generalized even further by introducing even factors of weakly symmetric digraphs. Later, a purely combinatorial approach to even factors was given by Gy. Pap and L. Szegő, who showed that the maximum even factor problem remains tractable in the class of hardly symmetric digraphs. The present paper shows a direct polyhedral way to derive weighted integer min-max formulae generalizing those previous results.

1 Introduction

Cunningham and Geelen [2,3] introduced the notion of *even factor* in digraphs as the edge set of vertex disjoint union of *dicycles* of even length and *dipaths*. Here dicycle (dipath) means directed cycle (directed path), while a cycle (path) can contain both forward and backward edges. The maximum cardinality even factor problem is NP-hard in general (Wang, see [3]) but there are special classes of digraphs where it can be solved.

In the paper digraphs are assumed to have no loops and no parallel edges. An edge of a digraph is called *symmetric* if the reversed edge is also in the edge set of the digraph. A digraph is symmetric if all its edges are symmetric, while a digraph is said to be *weakly symmetric* if its strongly connected components are symmetric. Thus weak symmetry means that the edges that belong to a dicycle are symmetric. Pap and Szegő [10] introduced the more general class of *hardly symmetric* digraphs where the problem remains tractable. A digraph is said to be hardly symmetric if the edges contained in an odd dicycle are symmetric.

Using an algebraic approach, Cunningham and Geelen proved a min-max formula for the maximum cardinality even factor problem in weakly symmetric digraphs [3]. Later, Pap and Szegő [10] proved a simpler formula for the same

* Research supported by the Hungarian National Foundation for Scientific Research, OTKA T037547 and OTKA N034040, and by European MCRTN Adonet, Contract Grant No. 504438.

problem by a purely combinatorial method and they also observed that the same proof and formula works for the hardly symmetric case.

As the unweighted problem is tractable only in special digraphs, the weight function also has to possess certain symmetries. If $G' = (V', A')$ is a weakly symmetric digraph and $c' : A' \rightarrow \mathbb{R}$ is a weight function s.t. $c'(uv) = c'(vu)$ if both uv and vu belong to A' , then the pair (G', c') is called *weakly symmetric*. Cunningham and Geelen considered a weighted generalization in [3] for weakly symmetric pairs. Using their unweighted formula and a primal-dual method they derived integrality of a polyhedron similar to the perfect matching polyhedron.

The present paper deals with a more general class of digraphs and weight functions which include the above mentioned cases. We show integrality of polyhedra related to even factors and a total dual integral system which generalizes among others the path-matching polyhedron of Cunningham and Geelen [4]. Our proofs use only polyhedral techniques, without relying on earlier weighted and unweighted results.

We conclude this section by stating the main result of the paper. To simplify the notation, the singleton $\{v\}$ will sometimes be denoted by v . For $x : V \rightarrow \mathbb{R}$ and $U \subseteq V$ we use the notation $x(U) = \sum_{v \in U} x(v)$ and we identify the functions $x : V \rightarrow \mathbb{R}$ with the vectors $x \in \mathbb{R}^V$. A family \mathcal{F} of subsets of V is said to be *laminar* if for any $X, Y \in \mathcal{F}$, at least one of $X \subseteq Y$, $Y \subseteq X$, or $X \cap Y = \emptyset$ holds.

Similarly to the definition of weakly symmetric pairs, a *pair* of a digraph $G' = (V', A')$ and a weight function $c' : A' \rightarrow \mathbb{R}$ is said to be *hardly symmetric* if G' is hardly symmetric and $c'(C) = c'(\overline{C})$ for every odd dicycle C of G' , where \overline{C} denotes the odd dicycle obtained from C by reversing the direction of its edges.

If $G = (V, A)$ is a directed graph and $U \subseteq V$, then $G[U]$ stands for the subgraph of G induced by U ; $\delta^{in}(U)$, $\delta^{out}(U)$ and $i(U)$ denote respectively the set of edges *entering*, *leaving* and *spanned* by U . Although the digraph is not indicated in these notations, it will always be clear from the context. The main result of the paper is as follows.

Theorem 1. *If $(G = (V, A), c)$ is a hardly symmetric pair and c is integral, then the optimal solution of the linear program*

$$\max cx \tag{1}$$

$$x \in \mathbb{R}^A, x \geq 0 \tag{2}$$

$$x(\delta^{in}(v)) \leq 1 \quad \text{for every } v \in V \tag{3}$$

$$x(\delta^{out}(v)) \leq 1 \quad \text{for every } v \in V \tag{4}$$

$$x(i(S)) \leq |S| - 1 \quad \text{for every } S \subseteq V, |S| \text{ odd} \tag{5}$$

is attained on an integer vector, and the dual of that linear program has an integer optimal solution $(\lambda_v^{in}, \lambda_v^{out}, z_S)$, where $\mathcal{F} = \{S : z_S > 0\}$ is a laminar family, and $G[S]$ is strongly connected and symmetric for each $S \in \mathcal{F}$.

The solution set of (2)-(3)-(4)-(5) is denoted by $\text{EF}(G)$. It is easy to see that for any digraph G , the integer solutions of $\text{EF}(G)$ are exactly the incidence

vectors of even factors of G . While this polyhedron is not integer in general, we can guarantee an integral optimum solution for the above class of digraphs and weights. The proof of Theorem 1 uses a direct polyhedral technique. First, dual

integrality is proved for hardly symmetric pairs, by considering a rational optimal dual solution which is extremal in some sense, and by altering it to reach integrality while optimality is maintained. Primal integrality can then be derived from this result. This technique resembles the way of proving integrality via the powerful notion of total dual integrality introduced by Edmonds and Giles [6]. The difference is that here we have dual integrality only for a restricted class of weight functions.

2 Preliminaries

The maximal strongly connected subgraphs of a digraph having 2-vertex-connected underlying undirected graph are called *blocks*. The following crucial observation on the structure of hardly symmetric digraphs was made by Z. Király [8].

Lemma 1 (Király). *Each block of a hardly symmetric digraph is either symmetric or bipartite.*

It follows that any odd dicycle is entirely spanned by some symmetric block. A dicycle C contained in a symmetric block is symmetric regardless of the parity of $|C|$. A similar statement is true for weight functions. A block is said to be *non-bipartite* if it contains an odd cycle, which by Lemma 1 implies that it contains an odd dicycle.

Lemma 2. *If (G, c) is a hardly symmetric pair and C is a dicycle contained in some symmetric non-bipartite block, then $c(C) = c(\overline{C})$.*

For a hardly symmetric digraph $G = (V, A)$, let $\mathcal{D}(G)$ denote the family of vertex-sets of its blocks, which partitions into the family of vertex-sets of bipartite blocks $\mathcal{B}(G)$ and the family of vertex-sets of non-bipartite (hence symmetric) blocks $\mathcal{S}(G)$. Let moreover $\mathcal{C}(G)$ be the family of vertex-sets of the strongly connected components of G .

In the following we characterize weight functions in hardly symmetric pairs. Clearly, the modification of the weight of edges which are not spanned by some strongly connected component preserves the hardly symmetric property. The same holds for the edges of $G[U]$ for every $U \in \mathcal{B}(G)$. The following lemma describes weights in non-bipartite blocks.

Lemma 3. *Let $(G = (V, A), c)$ be a hardly symmetric pair s.t. $\mathcal{S}(G) = \{V\}$. Then c is of the form*

$$c = \sum_{v \in V} b_v^{in} \chi_{\delta^{in}(v)} + \sum_{v \in V} b_v^{out} \chi_{\delta^{out}(v)} + \sum_{e \in A} b_e \chi_{\{e, \bar{e}\}}$$

for some reals b_v^{in} , b_v^{out} and b_e , where χ stands for the characteristic function.

Let $\mathcal{S}_*(G)$ denote the family of vertex-sets of the components of the hypergraph $(\cup\mathcal{S}(G), \mathcal{S}(G))$. It is easy to see that $\text{EF}(G)$ equals to the solution set of

$$x \in \mathbb{R}^A, \quad x \geq 0 \quad (6)$$

$$x(\delta^{in}(v)) \leq 1 \quad \text{for every } v \in V \quad (7)$$

$$x(\delta^{out}(v)) \leq 1 \quad \text{for every } v \in V \quad (8)$$

$$x(i(S)) \leq |S| - 1 \quad \text{for every } S \subseteq U, |S| \text{ odd}, U \in \mathcal{S}_*(G) . \quad (9)$$

Thus to prove Theorem 1, it is enough to consider this system.

3 Partial Dual Integrality

This section describes a general framework for obtaining primal integrality results based on the structure of the set of weight functions for which integral dual optimal solutions exist. We consider a bounded primal linear system of the form

$$\max\{cx : Ax \leq b, x \geq 0\} , \quad (10)$$

where A is a fixed $m \times n$ integer matrix, and b is a fixed integer vector. The dual system is

$$\min\{yb : yA \geq c, y \geq 0\} . \quad (11)$$

Let x_0 be a vertex of the primal polyhedron, and let $C \subseteq \mathbb{Z}^n$ be the set of integer weight vectors for which x_0 is optimal, and in addition there is an integral optimal dual solution. The strong duality theorem implies that cx_0 is an integer for every $c \in C$. Another easy observation is that C is closed under addition. Indeed, let $c_1, c_2 \in C$, and let y_1, y_2 be integral optimal dual solutions for c_1 and c_2 , respectively. Now x_0 and $y_1 + y_2$ satisfy the complementarity conditions for the weight vector $c_1 + c_2$, which means that x_0 is an optimal primal solution and $y_1 + y_2$ is an optimal dual solution for $c_1 + c_2$.

The following Lemma states that in a certain sense the reverse of the above observations is also true.

Lemma 4. *Let $C \subseteq \mathbb{Z}^n$ be a set of integer weight vectors that is closed under addition, and suppose that for every $c \in C$ there is an integral optimal solution y_c of the dual system (11). Then for every $c_0 \in C$ there is an optimal solution x_0 of the primal system (10) such that zx_0 is an integer for every $z \in C$ (and thus for every integer combination of vectors in C).*

Proof. Let $c_0 \in C$; we will find an x_0 with the above properties. Let z_0, z_1, \dots be an enumeration of the elements of C such that $z_0 = c_0$. We will construct a sequence of weight vectors c_0, c_1, \dots (all of them in C) such that there is a vertex of the polyhedron (10) that is optimal for all of these weight vectors. The general step is the following:

- Suppose that c_0, c_1, \dots, c_i are already defined, and let the face F_i be the set of points that are optimal primal solutions for c_i . The set of weight vectors for which the optimal primal solutions are a subset of F_i form an open set in \mathbb{R}^n . Thus there exists a positive integer k such that the optimal primal solutions for the weight vector $kc_i + z_{i+1}$ form a sub-face of F_i . Let $c_{i+1} := kc_i + z_{i+1}$.

Since we have $F_0 \supseteq F_1 \supseteq \dots$ for the sequence of faces, $\cap_{i=0}^{\infty} F_i$ is a non-empty face of the primal polyhedron (10). Let x_0 be an arbitrary vertex of this face; then x_0 is optimal for all of c_0, c_1, \dots . The strong duality theorem implies that $c_i x_0$ is an integer for every i , and therefore $z x_0$ is an integer for every integer combination z of those weight vectors. But the elements of C can be obtained as integer combinations of that type. \square

If $C = \mathbb{Z}^n$, then the above defined partial dual integrality specializes to the notion of *total dual integrality* introduced by Edmonds and Giles [6], which also extends to the case of unbounded polyhedra. “Total dual integral” is usually abbreviated as *TDI*.

It is well-known that the operation of replacing any inequality $ax \leq \beta$ of a TDI system by the equality $ax = \beta$ yields again a TDI system. This property also holds for partial dual integrality under some circumstances, which allows us to exploit the integrality of full dimensional systems while proving integrality results for faces of the original polyhedra.

Lemma 5. *We are given $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ and a set of integer weight functions $C \subseteq \mathbb{Z}^n$ which is closed under addition. Suppose that for each $c \in C$ s.t. the linear program $\max\{cx : Ax \leq b\}$ and its dual have finite optima, these are attained on integer vectors. Then the same holds for $\max\{cx : Ax \leq b, ax = \beta\}$ where $ax \leq \beta$ is a row of $Ax \leq b$ and $a \in C$.*

Proof. For $c \in C$,

$$\max\{cx : Ax \leq b, ax = \beta\} = \min\{yb + z\beta : y \geq 0, yA + za = c\} \quad (12)$$

and these finite values are attained on (the possibly fractional) x^*, y^*, z^* . Let $c' = c + Na$ where $N \in \mathbb{Z}$, $z^* + N > 0$. Then the optima

$$\max\{c'x : Ax \leq b, ax \leq \beta\} = \min\{yb + z\beta : y \geq 0, z \geq 0, yA + za = c'\} \quad (13)$$

are finite, since x^* and $y^*, z^* + N$ are feasible optimum solutions for them. Moreover, $z^* + N > 0$ implies that every optimum solution x for the maximum of (13) satisfies $ax = b$. Thus $c' \in C$ implies that these optima of (13) are attained on integer vectors $\tilde{x}, \tilde{y}, \tilde{z}$. Finally, $x = \tilde{x}$ and $y = \tilde{y}, z = \tilde{z} - N$ are integer optimum solutions for the maximum and for the minimum of (12). \square

4 A TDI System

The polyhedral results presented in Theorem 1 extend the results of the second named author in [9] where primal integrality is proved for a more restricted class

of set functions, using total dual integrality of a different linear system. Here we cite these total dual integrality results, and we will show how they follow from Theorem 1.

Let $G = (V, A)$ be a hardly symmetric digraph, and let us consider a partition of $\mathcal{D}(G)$ into a set \mathcal{S} of blocks and a set \mathcal{B} of blocks s.t. for any $U \in \mathcal{S}$, $G[U]$ is symmetric and for any $U \in \mathcal{B}$, $G[U]$ is bipartite. Note that these families are not necessarily identical with $\mathcal{S}(G)$ and $\mathcal{B}(G)$ since a block can be both bipartite and symmetric. A weight function $c : A \rightarrow \mathbb{R}$ is said to be *evenly symmetric* (w.r.t. some G , \mathcal{S} and \mathcal{B}) if $c(uv) = c(vu)$ for any edge uv spanned by some $U \in \mathcal{S}$. Let \mathcal{S}_* be the family of vertex-sets of the components of the hypergraph $(\cup \mathcal{S}, \mathcal{S})$. Evenly symmetric weight functions enable us to handle the symmetric edges spanned by the members of \mathcal{S} as undirected edges, since these oppositely directed edges have the same weight. So a *mixed graph* is considered with vertex set V and edge set $E = E_d \cup E_u$,

$$\begin{aligned} E_u &= \{\{u, v\} : uv, vu \in \cup_{U \in \mathcal{S}} i(U)\} , \\ E_d &= \{uv \in A : \{u, v\} \notin E_u\} , \end{aligned}$$

where uv stands for the directed edge with tail u and head v , and $\{u, v\}$ for the undirected edge with endpoints u and v . For this mixed graph and for some $v \in V$ and $U \subseteq V$, $d(v)$ denotes the set of (directed and undirected) edges adjacent to v , $\delta^{in}(U)$ denotes the set of directed edges entering U , $\delta^{out}(U) = \delta^{in}(V - U)$, and $i(U)$ denotes the set of (directed and undirected) edges spanned by U .

For $x \in \text{EF}(G)$, let $x^s \in \mathbb{R}^E$ be defined by $x^s(e) = x(vu) + x(uv)$ if $e = \{u, v\} \in E_u$, and $x^s(e) = x(uv)$ if $e = uv \in E_d$. In the following we give a TDI system which describes the polyhedron

$$\text{SEF}(G) = \{y \in \mathbb{R}^E : \exists x \in \text{EF}(G), y = x^s\} .$$

It is easy to see that the integral points of this polyhedron correspond to even factors (however, an integral point does not define a unique even factor).

Theorem 2. *For a hardly symmetric digraph $G = (V, A)$ and corresponding families \mathcal{S} and \mathcal{B} , $\text{SEF}(G)$ is the solution set of*

$$y \in \mathbb{R}^E, y \geq 0 \tag{14}$$

$$y(d(v)) \leq 2 \quad \text{for every } v \in V \tag{15}$$

$$y(i(S)) \leq |S| - 1 \quad \text{for every } S \subseteq U, |S| \text{ odd}, U \in \mathcal{S}_* \tag{16}$$

$$y(i(S)) + y(\delta^{in}(S)) \leq |S| \quad \text{for every } S \in \mathcal{U} \tag{17}$$

$$y(i(S)) + y(\delta^{out}(S)) \leq |S| \quad \text{for every } S \in \mathcal{U} \tag{18}$$

where $\mathcal{U} = \{U : \exists S \in \mathcal{S}_* \text{ s.t. } U \subseteq S\} \cup \{\{v\} : v \in V - \cup \mathcal{S}\}$.

This system is TDI. Moreover, if $c \in \mathbb{Z}^E$ is an integer vector, then there is an integer optimal solution $(\mu_v^d, \mu_S^i, \mu_S^{i,in}, \mu_S^{i,out})$ of the dual system s.t.

$$\mathcal{G} = \{S : \mu_S^i > 0\} \cup \{S : \mu_S^{i,in} > 0\} \cup \{S : \mu_S^{i,out} > 0\} \text{ is a laminar family, } (19)$$

$$\text{if } \mu_S^i > 0, \mu_T^{i,in} > 0, S \cap T \neq \emptyset, \text{ then } S \subseteq T, \quad (20)$$

$$\text{if } \mu_S^i > 0, \mu_T^{i,out} > 0, S \cap T \neq \emptyset, \text{ then } S \subseteq T, \text{ and} \quad (21)$$

$$\mu_S^{i,out} > 0, \mu_T^{i,in} > 0 \text{ implies } S \cap T = \emptyset. \quad (22)$$

If G is symmetric and we choose $\mathcal{S} = \mathcal{D}(G)$, then this specializes to $2M(G)$ where $M(G)$ is the *matching polyhedron* defined by

$$\begin{aligned} y &\in \mathbb{R}^E, y \geq 0 \\ y(d(v)) &\leq 1 \quad \text{for every } v \in V \\ y(i(S)) &\leq \frac{|S| - 1}{2} \quad \text{for every } S \subseteq V, |S| \text{ odd.} \end{aligned}$$

This system was proved to be a TDI system by Cunningham and Marsh [5]. Another well-known special case is when the strongly connected components are all bipartite, i.e. $\mathcal{B} = \mathcal{D}(G)$ is a valid choice. In this case

$$\begin{aligned} y &\in \mathbb{R}^E, y \geq 0 \\ y(\delta^{in}(v)) &\leq 1 \quad \text{for every } v \in V \\ y(\delta^{out}(v)) &\leq 1 \quad \text{for every } v \in V \end{aligned}$$

is obtained, which is TDI by *total unimodularity*. Also, we can see easily that any integer solution of this system is a vertex disjoint union of dipaths and even dicycles.

A more general special case is the *path-matching polyhedron* which was introduced by Cunningham and Geelen. For a more detailed description, the reader is referred to the paper of Cunningham and Geelen [4] and to the paper of Frank and Szegő [7]. In short, we have a digraph $G = (V, A)$ and a partition of V into sets T_1 , R and T_2 , such that T_1 and T_2 are stable sets, $G[R]$ is symmetric, no edge enters T_1 , and no edge leaves T_2 . Let $\mathcal{S} = \mathcal{D}(G)$. Then the system (14)-(15)-(16)-(17)-(18) defined on this graph specializes to the path-matching polyhedron

$$\begin{aligned} y &\in \mathbb{R}^E, y \geq 0 \\ y(d(v)) &\leq 2 \quad \text{for every } v \in R \\ y(i(S)) &\leq |S| - 1 \quad \text{for every } S \subseteq R, |S| \text{ odd} \\ y(i(S)) + y(\delta^{in}(S)) &\leq |S| \quad \text{for every } S \subseteq R \\ y(i(S)) + y(\delta^{out}(S)) &\leq |S| \quad \text{for every } S \subseteq R \\ y(\delta^{in}(v)) &\leq 1 \quad \text{for every } v \in T_2 \\ y(\delta^{out}(v)) &\leq 1 \quad \text{for every } v \in T_1 \end{aligned}$$

which was proved to be TDI by Cunningham and Geelen [4].

As we mentioned, Cunningham and Geelen considered a weighted version of the even factor problem in weakly symmetric digraphs [3]. Given a weakly symmetric pair $(G = (V, A), c)$ and equicardinal subsets V^{in} and V^{out} of V s.t. $A \subseteq \{uv : u \in V^{out}, v \in V^{in}\}$, they considered the problem of finding a minimum weight even factor x satisfying $x(\delta^{out}(v)) = 1$ for every $v \in V^{out}$ and $x(\delta^{in}(v)) = 1$ for every $v \in V^{in}$. It is proved in [3] that this is algorithmically equivalent to finding maximum weight even factors in weakly symmetric digraphs with respect to weakly symmetric weight functions. The primal-dual method of Cunningham and Geelen implies that the optimal solution of the linear program

$$\begin{aligned} & \min cx \\ & x \in \mathbb{R}^A, \quad x \geq 0 \\ & x(\delta^{in}(v)) = 1 \quad \text{for every } v \in V^{in} \\ & x(\delta^{out}(v)) = 1 \quad \text{for every } v \in V^{out} \\ & x(\delta^{in}(S)) \geq 1 \quad \text{for every } S \subseteq V^{in} \cap V^{out}, |S| \text{ odd} \\ & x(\delta^{out}(S)) \geq 1 \quad \text{for every } S \subseteq V^{in} \cap V^{out}, |S| \text{ odd} \end{aligned}$$

is attained on an integer vector and this also holds for its dual if c is an integer weight function. The interested reader can verify that this can be derived from Theorem 1 and Lemma 5.

The rest of the paper is organized as follows. In the next section, we derive the unweighted min-max formula of Pap and Szegő [10], while in the subsequent sections, the detailed proofs of Theorems 1 and 2 are presented.

5 Unweighted Min-Max Formula

In [3], Cunningham and Geelen derived weighted results for weakly symmetric pairs using their unweighted min-max formula and a primal-dual method. Here we follow an opposite direction and prove the unweighted formula of Pap and Szegő [10] as a consequence of Theorem 1. First, to better understand the analogy with older results, let us recall a non-standard form of the well-known Berge-Tutte formula. If $G = (V, E)$ is an undirected graph then $\text{odd}(G)$ denotes the number of connected components of G having an odd number of vertices, and $N_G(X) = \{v \in V - X : \exists u \in X, \{u, v\} \in E\}$.

Theorem 3 (Berge and Tutte). *If $G = (V, E)$ is an undirected graph, then the cardinality of a maximum matching of G is*

$$\min_{X \subseteq V} \frac{|V| + |N_G(X)| - \text{odd}(G[X])}{2}.$$

In a digraph $G = (V, A)$, we define $N_G^{out}(X) = \{v \in V - X : \exists u \in X, uv \in A\}$ and let $\text{odd}(G)$ denote the *number of strongly connected components of G with no entering arc (i.e. source components) that have an odd number of vertices*. Using this notation, Pap and Szegő [10] proved the following.

Theorem 4 (Pap and Szegő). If $G = (V, A)$ is a hardly symmetric digraph, then the maximum cardinality of an even factor is

$$\min_{X \subseteq V} |V| + |N_G^{out}(X)| - \text{odd}(G[X]) .$$

Proof. First we prove $\max \leq \min$. For a fixed X , any even factor of G has at most $|X| - \text{odd}(G[X])$ edges spanned by X , at most $|N_G^{out}(X)|$ edges leaving X , and at most $|V - X|$ edges having tail in $V - X$. Thus an even factor of G has cardinality at most $|V| + |N_G^{out}(X)| - \text{odd}(G[X])$.

To see $\max \geq \min$, we consider integer optimal primal and dual solutions of $\max_{e \in A} x(e)$ subject to (2)-(3)-(4)-(5). We assume that for the dual solution $(\lambda_v^{in}, \lambda_v^{out}, z_S)$, $\mathcal{F} = \{S : z_S > 0\}$ is a laminar family, and $G[S]$ is strongly connected and symmetric for each $S \in \mathcal{F}$. Such a dual solution exists by Theorem 1. The primal solution is an even factor, while the dual solution can be modified to prove our formula. Clearly, the dual solution is 0-1 valued, hence \mathcal{F} forms a subpartition of V . Let us choose this dual solution so that $|\{v \in \cup \mathcal{F} : \lambda_v^{in} + \lambda_v^{out} > 0\}|$ is as small as possible.

Claim. If $v \in \cup \mathcal{F}$, then $\lambda_v^{in} + \lambda_v^{out} = 0$.

Proof. If $z_S = 1$ and $\lambda_v^{in} = 1$ for some $v \in S$, then we could change the dual variables to $z_S = 0$ and $\lambda_u^{in} = 1$ for every $u \in S$, while the other variables remain unchanged, which contradicts our extremal choice. \square

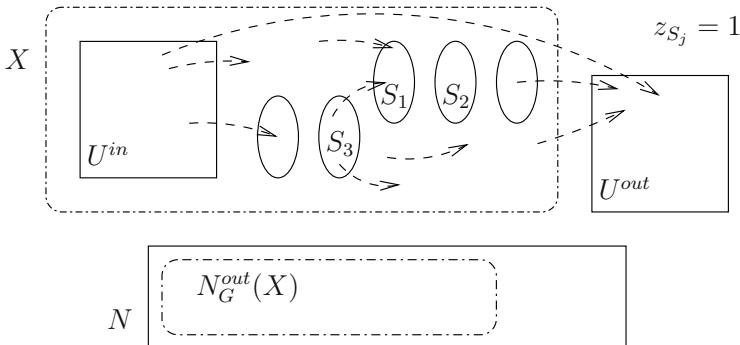


Fig. 1. The edges indicated by dashed line cannot occur.

From the dual solution $(\lambda_v^{in}, \lambda_v^{out}, z_S)$ we can define the set X for which $\max \geq \min$. Let $U^{in} = \{v : \lambda_v^{in} = 1, \lambda_v^{out} = 0\}$, $U^{out} = \{v : \lambda_v^{out} = 1, \lambda_v^{in} = 0\}$, $N = \{v : \lambda_v^{in} = \lambda_v^{out} = 1\}$ and $X = V - N - U^{out}$. Then $G[X - U^{in}]$ is composed of symmetric odd components, which are source components in $G[X]$ (see Fig.

1). Clearly, $N_G^{out}(X) \subseteq N$. Therefore the cardinality of a maximum even factor is at least

$$\begin{aligned} 2|N| + \sum_{z_S=1} (|S| - 1) + |U^{out}| + |U^{in}| &\geq \\ |N| + |V| - \text{odd}(G[X]) &\geq |V| + |N_G^{out}(X)| - \text{odd}(G[X]) . \end{aligned}$$

□

6 Structure of Hardly Symmetric Weights

Let $G' = (V', E')$ be a digraph. An *ear* of G' is a dicycle or a dipath (with different ends) of G' , while a *proper ear* of G' is a dipath (with different ends) of G' . The sequence $C_0, P_1, P_2, \dots, P_k$ is an *ear-decomposition* of G' if the following hold:

- C_0 is a dicycle called *initial dicycle*;
- every P_i is an ear;
- $C_0 \cup P_1 \cup \dots \cup P_{i-1}$ has exactly two common vertices with P_i , namely the ends of P_i , if P_i is a dipath, and has exactly one common vertex with P_i , if P_i is a dicycle;
- $C_0 \cup P_1 \cup \dots \cup P_k = G'$.

Similarly, the sequence $C_0, P_1, P_2, \dots, P_k$ is a *proper ear-decomposition* of G' if C_0 is a dicycle of length at least 2, called *initial dicycle*; every P_i is a proper ear; $C_0 \cup P_1 \cup \dots \cup P_{i-1}$ has exactly two common vertices with P_i , namely the ends of P_i ; and $C_0 \cup P_1 \cup \dots \cup P_k = G'$.

It is well-known that every strongly connected digraph G' has an ear-decomposition, moreover, every dicycle of G' can be the initial dicycle of this ear-decomposition. Similarly, a strongly connected digraph G' which is 2-vertex-connected in the undirected sense has a proper ear-decomposition, and every dicycle of G' of length at least 2 can be the initial dicycle of a proper ear-decomposition.

Proof (Proof of Lemma 1). Let $G = (V, A)$ be a hardly symmetric digraph with $\mathcal{D}(G) = \{V\}$. If G is bipartite, then we are done. Thus, G contains a (not necessarily directed) closed walk W having an odd number of edges (with multiplicity). W may have forward and backward edges; consider such a W containing minimum number of backward edges.

Our aim is to find a directed closed walk having an odd number of edges. If W has no backward edge, then we are done. If W has a backward edge uv , then by strong connectivity there is a dipath P from v to u . If P has an even number of edges, then P together with uv is an odd dicycle. Otherwise, uv can be replaced by P in W , and the number of backward edges decreases, contradicting the assumption. Thus there is a directed closed walk having an odd number of edges, which of course contains an odd dicycle C .

Claim. There is a sequence of symmetric digraphs $G_0 \subsetneq G_1 \subsetneq \dots \subsetneq G_l = G$ s.t. G_0 is a symmetric odd cycle, and for any two vertices s and t of G_i , G_i contains both odd and even length dipaths from s to t .

Proof. We consider a proper ear-decomposition of G with initial cycle C . Then C is odd, hence symmetric, and clearly there are both odd and even dipaths between any two vertices of C . Suppose, by induction, that a subgraph $G_i \subseteq G$ is built up by the proper ear-decomposition, G_i is symmetric, and there exist both odd and even dipaths in G_i between any two vertices of G_i . Then the edges of the next proper ear Q are contained in an odd dicycle, hence they are symmetric. It is easy to see that there are both odd and even dipaths between any two vertices of Q . \square

The claim completes the proof. \square

Proof (Proof of Lemma 2). We can assume that G is itself a symmetric non-bipartite block. Our goal is to prove that $c(C) = c(\overline{C})$ for every even cycle C of length at least 4 ($c(C) = c(\overline{C})$ automatically holds for a cycle of length 2). G has a proper ear decomposition C, P_1, P_2, \dots, P_i with initial cycle C . Since G is non-bipartite, there is a first ear, say P_j , whose addition violates the bipartite property. By 2-vertex-connectivity, $C \cup P_1 \cup P_2 \cup \dots \cup P_{j-1}$ contains vertex disjoint (all the inner and end vertices are different) paths (in the undirected sense) from the endpoints of P_j to C ; let these be Q_1 and Q_2 . Now $C \cup Q_1 \cup Q_2 \cup P_j$ is non-bipartite, and it has two odd cycles containing P_j . Using the symmetry of G and the property that on odd dicycles the weight-sum is the same in the two directions, this yields that $c(C) = c(\overline{C})$. \square

Proof (Proof of Lemma 3). Clearly, $\chi_{\delta^{in}(v)}$, $\chi_{\delta^{out}(v)}$ and $\chi_{\{e, \overline{e}\}}$ are hardly symmetric weight functions. Fix a root $s \in V$. If P and Q are dipaths from s to t , then $c(P) - c(\overline{P}) = c(Q) - c(\overline{Q})$ by Lemma 2, where \overline{P} is the dipath obtained by reversing the direction of the edges of P . Thus we can define $y : V \rightarrow \mathbb{R}$ with $y(t) = c(P) - c(\overline{P})$ where P is any $s - t$ dipath. Moreover, it is easy to see that $b = c - \sum_{v \in V} y(v)\chi_{\delta^{in}(v)}$ is a symmetric weight function i.e. $b(e) = b(\overline{e})$. Using y and b we can obtain the desired decomposition of c . \square

7 Proof of Theorem 1

Proof (Proof of Theorem 1). First we prove dual integrality. Let $\lambda_v^{in}, \lambda_v^{out} \geq 0$, $z_S \geq 0$ be a rational optimal dual solution (for $v \in V$ and $S \subseteq U, U \in \mathcal{S}_*(G)$, $|S|$ odd). We choose a positive integer k (which is fixed throughout the proof) s.t. $\tilde{\lambda}_v^{in} = k\lambda_v^{in}$, $\tilde{\lambda}_v^{out} = k\lambda_v^{out}$, $\tilde{z}_S = kz_S$ are all integers. For fixed k , let us choose moreover this solution so that the vector $(K = \sum_{v \in V} \tilde{\lambda}_v^{in} + \sum_{v \in V} \tilde{\lambda}_v^{out}, L = \sum_S \tilde{z}_S |S|^2)$ is lexicographically as large as possible. For optimal solutions and a fixed k , K and L are bounded from above. We show that under these conditions, the dual solution is almost integer and it can be transformed easily into an integer optimal one. The steps of the proof are motivated by the work of Balas and Pulleyblank [1].

Claim. $\mathcal{F} = \{S : z_S > 0\}$ is a laminar family.

Proof. If \mathcal{F} was not laminar, then the following simple uncrossing technique could be applied. Suppose that $z_S, z_T > 0$, and none of $S \cap T$, $S - T$, $T - S$ is empty. If $|S \cap T|$ is odd, then we can decrease z_S and z_T , and increase $z_{S \cap T}$ and $z_{S \cup T}$ by $\frac{1}{k}$, in which case K does not change and L increases, which is a contradiction. If $|S \cap T|$ is even, then we can decrease z_S and z_T by $\frac{1}{k}$, increase z_{S-T} and z_{T-S} by $\frac{1}{k}$ and increase λ_v^{in} and λ_v^{out} by $\frac{1}{k}$ if $v \in S \cap T$, so that K increases, which is a contradiction. \square

For a graph G and a laminar family \mathcal{F} , let $G \times \mathcal{F}$ denote the graph obtained from G by shrinking the maximal members of \mathcal{F} . For any $S \subseteq V$ s.t. $\{S\} \cup \mathcal{F}$ is again a laminar family, we let $\mathcal{F}[S] = \{U \in \mathcal{F} : U \subsetneq S\}$. Thus $G[S] \times \mathcal{F}[S]$ is the graph obtained from $G[S]$ by shrinking the maximal elements of \mathcal{F} properly contained in S . Let $A^=$ denote the set of tight edges, i.e. $A^= = \{uv \in A : \lambda_u^{out} + \lambda_v^{in} + \sum_{u,v \in S} z_S = c(uv)\}$ and let $G^= = (V, A^=)$.

Claim. For every $S \in \mathcal{F}$, $G^=[S] \times \mathcal{F}[S]$ is strongly connected. As a consequence, $G^=[S]$ is strongly connected for any $S \in \mathcal{F}$.

Proof. If $G^=[S] \times \mathcal{F}[S]$ is not strongly connected, then it has a strongly connected component U with an odd number of vertices, moreover, S has a partition into (possibly empty) sets U^{out} , U and U^{in} , s.t. any strongly connected component of $G^=[S] \times \mathcal{F}[S]$ is contained in some of the partition classes, and if $uv \in A^=$ connects two different classes, then one of the following three possibilities holds: $u \in U$ and $v \in U^{in}$; or $u \in U^{out}$ and $v \in U$; or $u \in U^{out}$ and $v \in U^{in}$. Now we can modify the dual solution in the following way:

$$\begin{aligned} z'_W &= \begin{cases} z_W - \frac{1}{k} & \text{if } W = S \\ z_W + \frac{1}{k} & \text{if } W = U \\ z_W & \text{otherwise} \end{cases} \\ (\lambda_v^{in})' &= \begin{cases} \lambda_v^{in} + \frac{1}{k} & \text{if } v \in U^{in} \\ \lambda_v^{in} & \text{otherwise} \end{cases} \\ (\lambda_v^{out})' &= \begin{cases} \lambda_v^{out} + \frac{1}{k} & \text{if } v \in U^{out} \\ \lambda_v^{out} & \text{otherwise.} \end{cases} \end{aligned}$$

This step yields a new dual solution with bigger K which is a contradiction. \square

Claim. For any $S \in \mathcal{F}$, $G^=[S] \times \mathcal{F}[S]$ is non-bipartite.

Proof. For contradiction, we choose an S for which $G^=[S] \times \mathcal{F}[S]$ is bipartite. Thus S has a bipartition $\{S_1, S_2\}$ so that any member of $\mathcal{F}[S]$ is a subset of some S_i , and $G^=[S_i] \times \mathcal{F}_i$ does not contain tight edges, where \mathcal{F}_i is the family of maximal members of $\{U \in \mathcal{F} : U \subseteq S_i\}$. We can assume moreover that $|S_1 - \cup_{S \in \mathcal{F}_1} S| + |\mathcal{F}_1| < |S_2 - \cup_{S \in \mathcal{F}_2} S| + |\mathcal{F}_2|$. We apply the following modification of the dual solution:

$$z'_W = \begin{cases} z_W + \frac{1}{k} & \text{if } W \in \mathcal{F}_2 \\ z_W - \frac{1}{k} & \text{if } W \in \mathcal{F}_1 \text{ or } W = S \\ z_W & \text{otherwise} \end{cases}$$

$$(\lambda_v^{in})' = \begin{cases} \lambda_v^{in} + \frac{1}{k} & \text{if } v \in S_1 \\ \lambda_v^{in} & \text{otherwise} \end{cases}$$

$$(\lambda_v^{out})' = \begin{cases} \lambda_v^{out} + \frac{1}{k} & \text{if } v \in S_1 \\ \lambda_v^{out} & \text{otherwise.} \end{cases}$$

A dual solution is obtained, with larger K , which leads to a contradiction. \square

The notation $a \equiv b$ is used if two integers a and b are congruent modulo k (i.e. $a - b$ is divisible by k). For the equivalence class of a under the modulo k equivalence relation we write \bar{a} .

Claim. For any $S \in \mathcal{F}$, $\tilde{\lambda}_v^{in} \equiv \tilde{\lambda}_u^{in}$ and $\tilde{\lambda}_v^{out} \equiv \tilde{\lambda}_u^{out}$ whenever $u, v \in S$.

Proof. For contradiction, we can choose a minimal $S \in \mathcal{F}$ for which the statement does not hold. Now $G^=[S]$ is symmetric, $G^=[S] \times \mathcal{F}[S]$ is strongly connected, symmetric and non-bipartite. So let uv and vu be edges of $G^=[S] \times \mathcal{F}[S]$ where $u, v \in S$. We let $\tilde{\lambda}_u^{out} \equiv a$, $\tilde{\lambda}_u^{in} \equiv b$, $\tilde{\lambda}_v^{out} \equiv c$ and $\tilde{\lambda}_v^{in} \equiv d$ (in short, u is of type (a, b) and v is of type (c, d)). Using the statement for the maximal members of $\mathcal{F}[S]$, and the fact that $G^=[S] \times \mathcal{F}[S]$ is strongly connected, we get that every vertex of S is either of type (a, b) or of type (c, d) , and the vertices in a maximal member of $\mathcal{F}[S]$ are of the same type. In addition, every edge of $G^=[S] \times \mathcal{F}[S]$ has end-vertices of both types. But $G^=[S] \times \mathcal{F}[S]$ is non-bipartite, hence $a \equiv c$ and $b \equiv d$. \square

We let \mathcal{F}_{\max} denote the maximal members of \mathcal{F} . As a consequence we get the following.

Claim. If $S \in \mathcal{F} - \mathcal{F}_{\max}$, then z_S is integer.

From now on, the property that (K, L) is lexicographically as large as possible will not be used; the actual solution can be easily transformed to an integer one. To this end, the number of nonzero modulo k remainders of the variables is decreased repeatedly until the desired integer optimal dual solution is obtained.

It can be seen that if λ_v^{out} and λ_v^{in} are integers for every vertex v then we are at an integer solution. Let us consider an integer $a \not\equiv 0$ s.t. $-a \equiv \tilde{\lambda}_v^{out}$ or $a \equiv \tilde{\lambda}_v^{in}$ for some $v \in V$, and define $U^{in} = \{v \in V : \tilde{\lambda}_v^{in} \equiv a\}$, $U^{out} = \{v \in V : \tilde{\lambda}_v^{out} \equiv -a\}$, $Z^{in} = \{S : S \in \mathcal{F}_{\max}, \forall v \in S \tilde{\lambda}_v^{in} \equiv a \text{ and } \tilde{\lambda}_v^{out} \not\equiv -a\}$ and $Z^{out} = \{S : S \in \mathcal{F}_{\max}, \forall v \in S \tilde{\lambda}_v^{out} \equiv -a \text{ and } \tilde{\lambda}_v^{in} \not\equiv a\}$. By symmetry we can assume that $|U^{in}| - \sum_{S \in Z^{in}} (|S| - 1) \geq |U^{out}| - \sum_{S \in Z^{out}} (|S| - 1)$. Then we can apply the following change in the dual solution:

$$\begin{aligned}
(\lambda_v^{in})' &= \begin{cases} \lambda_v^{in} - \eta & \text{if } v \in U^{in} \\ \lambda_v^{in} & \text{otherwise} \end{cases} \\
(\lambda_v^{out})' &= \begin{cases} \lambda_v^{out} + \eta & \text{if } v \in U^{out} \\ \lambda_v^{out} & \text{otherwise} \end{cases} \\
z'_W &= \begin{cases} z_W + \eta & \text{if } S \in Z^{in} \\ z_W & \text{otherwise} \end{cases} \\
z'_S &= \begin{cases} z_S - \eta & \text{if } S \in Z^{out} \\ z_S & \text{otherwise.} \end{cases}
\end{aligned}$$

The value η can be chosen to be a positive integer multiple of $\frac{1}{k}$ so that the value $\left| \left\{ \overline{\tilde{\lambda}_v^{in}} : v \in V, \tilde{\lambda}_v^{in} \neq 0 \right\} \cup \left\{ \overline{-\tilde{\lambda}_v^{out}} : v \in V, \tilde{\lambda}_v^{out} \neq 0 \right\} \right| + |\{S \in \mathcal{F} : \tilde{z}_S \neq 0\}|$ decreases by at least one. This means that after finitely many applications of this step we obtain an integer optimal dual solution.

Next we prove primal integrality. We apply Lemma 4 with C being the set of integer hardly symmetric weight functions. Clearly, this set is additive and contains the vectors $\chi_{\{e\}}$ for $e \notin \cup_{U \in \mathcal{S}(G)} i(U)$ and $\chi_{\{e, \bar{e}\}}$ for $e \in \cup_{U \in \mathcal{S}(G)} i(U)$. Thus there is an optimal solution x s.t. $x(e)$ is integer (0 or 1) if $e \notin \cup_{U \in \mathcal{S}(G)} i(U)$ and $x(e) + x(\bar{e})$ is integer (0, 1 or 2) if $e \in \cup_{U \in \mathcal{S}(G)} i(U)$. An even factor of weight at least cx can easily be constructed from x . \square

8 Proof of Theorem 2

Proof (Proof of Theorem 2). Let $G = (V, A)$ be a hardly symmetric digraph (with given \mathcal{B} and \mathcal{S}), and c an integral evenly symmetric weight function. The system $\max\{cx : x \in \text{EF}(G)\}$ has an integer dual optimal solution $(\lambda_v^{in}, \lambda_v^{out}, z_S)$ with the properties described in Theorem 1. From this, we will obtain a dual optimal solution $(\mu_v^d, \mu_S^i, \mu_S^{i,in}, \mu_S^{i,out})$ of $\{(14) - (15) - (16) - (17) - (18)\}$, satisfying the conditions of Theorem 2. During the construction, we consider the union of the two systems, and we start with $\mu_v^d = \mu_S^i = \mu_S^{i,in} = \mu_S^{i,out} = 0$ for every v, S , and end with $\lambda_v^{in} = \lambda_v^{out} = z_S = 0$ for every v, S . The steps of the construction are the following:

1. For $v \notin \cup \mathcal{S}$, let $\mu_v^{i,in} := \lambda_v^{in}$, $\mu_v^{i,out} := \lambda_v^{out}$, and set $\lambda_v^{in} = \lambda_v^{out} := 0$. For every $S \subseteq V$, we let $\mu_S^i := z_S$, and set $z_S := 0$.
2. Let us consider a set $T \in \mathcal{S}_*(G)$, and let T_1, \dots, T_k denote the strongly connected components of $G = [T]$. Then $G = [T_i]$ is symmetric for every i . As a consequence, $\lambda_v^{in} - \lambda_v^{out}$ is constant inside a set T_i . We also know that if $S \in \mathcal{F}$ and $S \cap T \neq \emptyset$, then S is subset of some T_i .
3. For every $v \in T$, let $\mu_v^d := \min\{\lambda_v^{in}, \lambda_v^{out}\}$, and decrease λ_v^{in} and λ_v^{out} by μ_v^d . Thus for every T_i either $\lambda_v^{in} = 0$ and λ_v^{out} is constant for every $v \in T_i$, or vice versa.
4. We can assume that $\lambda_v^{in} > 0$ for some $v \in T$. Let T_j be the strongly connected component on which λ_v^{in} is maximal. We increase $\mu_{T_j}^{i,in}$ by one, and decrease λ_v^{in} by one for every $v \in T_j$. There is no edge $uv \in G = [T]$ which enters

T_j , since then vu would also be in $G^=[T]$ by the maximality of λ_v^{in} , which contradicts the fact that T_j is a strongly connected component. Thus the obtained solution is feasible, and it is easy to see that it is also dual optimal. In addition, no edge disappears from $G^=$.

5. We repeat the above steps 2.–4. with the new $G^=$, until we obtain $\lambda_v^{in} = \lambda_v^{out} = 0$ for every v . The laminarity properties described in Theorem 2 are ensured by the fact that edges are never removed from $G^=$, so the strongly connected components can only grow.

□

Acknowledgements

The authors wish to thank András Frank, Zoltán Király, Gyula Pap and László Szegő for useful discussions on the topic.

References

1. E. Balas and W.R. Pulleyblank, *The perfectly matchable subgraph polytope of an arbitrary graph*, Combinatorica, 9 (1989), 321–337.
2. W.H. Cunningham, *Matchings, matroids, and extensions*, Math. Programming, Ser B., (2001).
3. W.H. Cunningham and J.F. Geelen, *Vertex-disjoint directed paths and even circuits*, manuscript.
4. W.H. Cunningham and J.F. Geelen, *The Optimal Path-Matching Problem*, Combinatorica, 17 (1997), 315–336.
5. W.H. Cunningham and A.B. Marsh III, *A primal algorithm for optimum matching*, Math. Programming Stud., 8 (1978), 50–72.
6. J. Edmonds and R. Giles, *A min-max relation for submodular functions on graphs*, Ann. Discrete Math. 1 (1977), 185–204.
7. A. Frank and L. Szegő, *A Note on the Path-Matching Formula*, Journal of Graph Theory, Vol 41, Issue 2 (2002), 110–119.
8. Z. Király, *personal communication*, (2002).
9. M. Makai, *A polyhedral approach to even factors*, EGRES Tech. Rep. 2003-05.
10. Gy. Pap and L. Szegő, *On the Maximum Even Factor in Weakly Symmetric Graphs*, to appear in J. Comb. Theory Series B.

Optimizing over Semimetric Polytopes

Antonio Frangioni¹, Andrea Lodi², and Giovanni Rinaldi³

¹ Dipartimento di Informatica, Univertità di Pisa,
Largo B. Pontecorvo 1, 56127 Pisa, Italy
frangio@di.unipi.it

² D.E.I.S., University of Bologna,
Viale Risorgimento 2, 40136 Bologna, Italy
alodi@deis.unibo.it

³ Istituto di Analisi dei Sistemi ed Informatica “Antonio Ruberti” del CNR,
Viale Manzoni 30, 00185 Roma, Italy
rinaldi@iasi.cnr.it

1 Introduction

Let $G = (V, E)$ be a complete graph. Then the *semimetric polytope* $\mathcal{M}(G)$ associated with G is defined by the following system of inequalities

$$\left. \begin{array}{l} x_{ij} + x_{ik} + x_{jk} \leq 2 \\ x_{ij} - x_{ik} - x_{jk} \leq 0 \\ -x_{ij} + x_{ik} - x_{jk} \leq 0 \\ -x_{ij} - x_{ik} + x_{jk} \leq 0 \end{array} \right\} \quad \text{for all distinct } i, j, k \in V, \quad (1)$$

called the *triangle inequalities*. The paper deals with the problem of finding efficient algorithms to optimize a linear function over such a polytope.

We briefly mention two reasons to seek for an efficient way to optimize a linear function over $\mathcal{M}(G)$.

1. The polytope $\mathcal{M}(G)$, for $|V| > 4$, properly contains the *cut polytope* associated with G (see, e.g., [7] for the details). Thus, if an edge cost function $c \in \mathbb{R}^E$ is given, maximizing c over $\mathcal{M}(G)$ produces an upper bound on the maximum c -value cut of G , which can be exploited in branch and bound or branch and cut schemes for solving max-cut to optimality. Actually, in all the computational studies concerning instances of max-cut for very large sparse graphs based on a branch and cut scheme, the only relaxation exploited is $\mathcal{M}(G)$, or, more precisely, its projection that will be described shortly later. Consequently, most of the computation time for finding a maximum cut is spent into a (possibly long) series of linear optimizations over $\mathcal{M}(G)$.
2. If an edge capacity function $c \in \mathbb{R}^E$ and an edge demand function $d \in \mathbb{R}^E$ are given, the existence of a feasible *miflow* in the network defined by G , c , and d is established by the Japanese Theorem (see, e.g., [15]). According to this theorem, a feasible miflow exists if and only if $\mu \cdot (c - d) \geq 0$ holds for every metric μ on V , i.e., for every point of the cone defined by all the homogeneous equations of (1). It is not hard to see that this is equivalent

to the condition $\min\{(c - d)x \mid x \in \mathcal{M}(G)\} \geq 0$. In some approaches to network design problems [2,6] such a feasibility problem has to be solved several times. Again, this calls for an effective solution algorithm.

Although these problems are just standard linear programs with a polynomial number of constraints ($4\binom{|V|}{3}$), they turn out to be surprisingly difficult to solve with standard LP tools such as the simplex method or interior-point methods, even if state-of-the-art software is used. It is therefore of considerable interest to develop alternative algorithmic techniques that make it possible to compute (even with some degree of approximation) optimal primal and dual solutions of these LP's, faster than it is currently possible with standard methods.

An alternative technique is the Lagrangian approach where one “dualizes” all the triangle inequalities and leaves only the upper and lower bounds on the variables (which are actually redundant in the system (1)) as explicit constraints. Such an approach has been successfully applied, e.g., in [3], where a subgradient-type approach is used to solve the Lagrangian dual.

We propose to extend this approach in two ways. First, we dualize only a subset of the triangle inequalities, leaving, as explicit constraints, all the inequalities associated with the triangles of G that have a selected node in common. Such a system of inequalities defines the *rooted semimetric polytope*. We show later how a linear problem defined on this polytope can be solved efficiently. Then, we test the use of a bundle-type algorithm [8] as an alternative to subgradient-type approaches to solve the Lagrangian dual; since subgradient-type approaches can be seen as “special cases” of bundle-type ones [1], this alternative can be considered an extension. We show that, in most cases, bundle-type approaches, in comparison with subgradient-type ones, either produce primal and/or dual solutions of substantially higher accuracy, or reduce the running times significantly, or achieve both these results. The best Lagrangian approach we implemented is shown to obtain, on a large set of instances, primal and dual solutions of accuracy comparable with that produced by standard LP algorithms but in a small fraction of their computation time.

2 Optimizing over the Rooted Semimetric Polytope

If G is not complete, the semimetric polytope associated with G can be obtained by projecting the set defined by (1) into \mathbb{R}^E . The resulting polytope is defined as follows. Let \mathcal{C} be the set of all chordless cycles of G and \bar{E} the subset of the edges of G that do not belong to a 3-edge cycle (a *triangle*) of G . The following system of linear inequalities

$$x(C \setminus F) - x(F) \leq |F| - 1 \quad F \subseteq C \text{ with } |F| \text{ odd and } C \in \mathcal{C} \quad (2)$$

$$0 \leq x_e \leq 1 \quad e \in \bar{E} \quad (3)$$

defines the *semimetric polytope* $\mathcal{M}(G)$ associated with G . Obviously, $\mathcal{M}(G)$ contains the cut polytope associated with G and every its integral point is the incidence vector of a cut of G . The inequalities (2) are called *cycle inequalities*.

Although exponentially many, they can be separated in polynomial time (see [5]).

Let r be a selected node of G that will be denoted as the *root*. Without loss of generality, assume that node r is adjacent to every other node of G . If this is not the case, one can make r adjacent to each other node by adding new edges to the graph and by extending the edge weight vector c assigning a zero weight to the new edges. A cut in the new graph is readily associated with a cut in the original graph and the two cuts have the same weight.

Let us partition the set \mathcal{C} into two subsets: the set \mathcal{C}_r of the chordless cycles that contain node r and its complement in \mathcal{C} . Under the above assumption, the edge set \bar{E} is empty and \mathcal{C}_r contains only triangles. The *rooted semimetric polytope* $\mathcal{M}^r(G)$ associated with G is defined by the subsystem of the cycle inequalities defined by \mathcal{C}_r , i.e.,

$$\left. \begin{array}{l} x_{ri} + x_{rj} + x_{ij} \leq 2 \\ x_{ri} - x_{rj} - x_{ij} \leq 0 \\ -x_{ri} + x_{rj} - x_{ij} \leq 0 \\ -x_{ri} - x_{rj} + x_{ij} \leq 0 \end{array} \right\} \quad \text{for all } ij \in E^r, \quad (4)$$

where E^r is the edge set of the subgraph of G induced by $V^r = V \setminus \{r\}$.

Despite the fact that the system (4) has much less inequalities than (1), it shares with (1) the property that it is satisfied by the incidence vector of any cut of G , while every its integral solution is the incidence vector of a cut of G . Therefore, the system (4) along with the integrality requirements provides an integer linear programming formulation for the max-cut problem. Thus, the optimization of $c \cdot x$ over $\mathcal{M}^r(G)$ yields an upper bound on the value of an optimal cut.

The system (4) has only $4|E^r|$ inequalities, but optimizing over the rooted semimetric polytope with a general purpose linear optimizer, although not as difficult as optimizing over $\mathcal{M}(G)$, is not as easy as one may expect. We first give a characterization of the vertices of $\mathcal{M}^r(G)$, then we outline a method to effectively solve the problem exploiting this characterization.

For any two disjoint subsets W and Z of V , the notation $(W : Z)$ stands for the set of edges with one endpoint in W and the other in Z . For a node set $W \subseteq V$, by $E(W)$ denote the set of all the edges in E with both the endpoints in W . If $Z = V \setminus W$, then the edge set $(W : Z) = (W : V \setminus W)$ is a *cut* of G and is also denoted by $\delta(W)$.

Definition 1. A *semitcut* of G is an ordered pair $\Omega = (F, J)$ of disjoint subsets of the edges of G satisfying the following condition: there exists an ordered pair (S, T) of disjoint subsets of V , with $r \in S$, such that

- (a) $F = (S : T) \cup H$, where $H \subseteq E(V \setminus (S \cup T))$, and
- (b) $J = \delta(S \cup T)$.

The edges in F are called regular, while those in J are called weak.

If $S \cup T = V$, then the set of weak edges is empty and the semicut is the ordered pair of a cut of G and of the empty set. Since a semicut is fully specified by the ordered pair of node sets (S, T) and by the edge set H , it will be denoted by $\Delta(S, T, H)$.

With a semicut $\Omega = \Delta(S, T, H)$ a representative vector $\psi^\Omega \in \mathbb{R}^E$ is associated, where for each $e \in E$ the component ψ_e^Ω is equal to 1 if e is a regular edge, is equal to $1/2$ if it is a weak edge, and is equal to 0 otherwise. If $S \cup T = V$, then H is empty and the characteristic vector of Ω is the incidence vector of a cut, namely, the cut $\delta(S)$. Conversely, the incidence vector of a cut $\delta(W)$ is the representative vector of the semicut $\Delta(W, V \setminus W, \emptyset)$. The *weight* of a semicut Ω is given by the inner product $c \cdot \psi^\Omega$.

A semicut of G is called *extreme* if its representative vector cannot be written as a convex combination of representative vectors of distinct semicuts of G . Not all semicuts are extreme. A characterization of the extreme semicuts and of the vertices of $\mathcal{M}^r(G)$ is given by the following theorems.

Theorem 1. *A semicut $\Omega = \Delta(S, T, H)$ is not extreme if and only if $U = V \setminus (S \cup T)$ is nonempty and there is a connected component $G_i = (U_i, E_i)$ of the subgraph of G induced by U for which $E_i \cap H$ is a cut of G_i .*

Corollary 1. *In a complete graph of n nodes the number of extreme semicuts for which $r \in S$ is given by*

$$2^{n-1} + \sum_{k=3}^{n-1} \binom{n-1}{k} (2^{\binom{k}{2}} - 2^{k-1}) 2^{n-k-1}$$

Theorem 2. *The set of vertices of $\mathcal{M}^r(G)$ coincides with the set of representative vectors of all the extreme semicuts of G for which $r \in S$.*

Let us now turn to the solution of

$$\max\{c \cdot x \mid x \in \mathcal{M}^r(G)\} \tag{5}$$

Consider a capacitated directed graph with node set $\{r\} \cup U \cup U'$ defined as follows. For each node $i \in V^r$ we associate the two nodes i and i' belonging to U and U' , respectively. Two opposite uncapacitated arcs connect r with every node in $U \cup U'$. With each edge ij ($i < j$) of E^r we associate two arcs (i, j') and (j, i') if $ij \in E_+^r = \{ij \in E^r \mid c_{ij} > 0\}$, and the two arcs (i, j) and (i', j') if $ij \in E_-^r = \{ij \in E^r \mid c_{ij} \leq 0\}$. The capacity of these arcs is $2|c_{ij}|$. Finally, consider the following minimum cost flow problem (MCFP) associated with this directed graph:

$$\min \sum_{i \in V^r} (u_{ri} + u_{ri'}) + \sum_{ij \in E_+^r} (v_{ij'} + v_{ji'})$$

subject to

$$\left. \begin{aligned} & u_{ri} + \sum_{ij \in E_+^r} v_{ij'} + \sum_{\substack{ki \in E_-^r \\ k < i}} w_{ki} - \sum_{\substack{ki \in E_-^r \\ k > i}} w_{ik} - q_{ir} = d_i \\ & -u_{ri'} - \sum_{ij \in E_+^r} v_{ji'} - \sum_{\substack{ki \in E_-^r \\ k < i}} w_{k'i'} + \sum_{\substack{ki \in E_-^r \\ k > i}} w_{i'k'} + q_{i'r} = -d_i \\ & v_{ij'} \leq 2|c_{ij}| \\ & v_{ji'} \leq 2|c_{ij}| \\ & w_{ij} \leq 2|c_{ij}| \\ & w_{i'j'} \leq 2|c_{ij}| \end{aligned} \right\} i \in V^r$$

$$u, v, w, q \geq 0 ,$$

where

$$d_i = c_{ri} + \sum_{ij \in E^r} c_{ij} - 2 \sum_{\substack{hi \in E_-^r \\ h < i}} c_{hi}. \quad (6)$$

Then we have the following

Theorem 3. *The optimal objective function value of MCFP equals twice the optimal value of (5).*

A sketch of the proof of this theorem is given in the Appendix. In addition, we can show that an optimal solution of (5) can be easily obtained from an optimal dual solution of MCFP, which in turn can be obtained from a primal optimal solution by exploiting the semicut structure. The characterization of the solutions of (5) as semicuts is useful also to devise an efficient algorithm for ranking all the solutions of (5) in the spirit of the scheme given, e.g., in [17] for ranking the s - t cuts in a graph.

Several polynomial time algorithms are available to solve MCFP. In particular, Cost Scaling algorithms run in $O(n^2 m \log nC)$ time, where n is the number of nodes of the graph, m the number of arcs, and C the largest absolute value of the components of the cost vector. Since in our case C is equal to 1, Cost Scaling algorithms solve (5) in strongly polynomial time.

We have performed extensive computational tests on several large-scale instances. In Table 1 we report a brief outcome of our testing on *clique* graphs, *planar* graphs, *simplex* graphs and *toroidal 2D and 3D-grid* graphs for spin glass problems, with either ± 1 or Gaussian costs (costs in \mathbb{R} drawn from a Gaussian distribution), for a total of 39 instances. The name of each instance begins with “c”, “p”, “s”, “g2-pm”, “g2-g”, and “g3-g”, respectively, the rest of the name gives the size of the node set. We tested different algorithms for MCFP implemented under the *MCFClass* interface [9], as well as with a general purpose LP solver. Specifically, we tested ILOG-Cplex 8.1 Network (CPXNET), the min-cost

flow network simplex implementation in [14] (ZIB) and the Cost Scaling algorithm in the implementation of [11] (CS2), and the general purpose LP solver ILOG-Cplex 8.1 Barrier directly applied to (5). We do not report on the performance of general purpose Primal Simplex and Dual Simplex algorithms as prior tests have shown that for this kind of instances they run much slower than Barrier. Computing times are in seconds on a Pentium M 1.6 GHz notebook, and each entry is an average over three different instances.

Name	directed graph		CPXNET time	ZIB time	CS2 time	ILOG-Cplex 8.1, Barrier			speed-up
	#nodes	# arcs				# var.s	# cons.s	time	
c150	299	22278.4	0.1	0.1	0.0	10990.2	43364.8	1.0	69.2
c500	999	246363.2	3.2	7.6	0.3	122682.6	488734.4	24.8	82.6
p10000	19999	96760.0	1.7	4.9	0.8	38380.7	113528.0	2.6	3.5
p20000	39999	193516.7	5.3	11.5	2.5	76758.0	227041.3	5.8	2.4
s39711	79421	603219.3	256.1	336.2	9.7	261899.7	888758.7	206.7	21.2
g2-pm22500	44999	179988.0	2.5	2.8	2.1	67495.0	179984.0	4.7	2.2
g2-pm62500	124999	499988.0	10.2	13.5	8.8	187495.0	499984.0	24.2	3.0
g3-pm64000	127999	639984.0	80.8	68.2	9.5	255993.0	767976.0	757.7	79.6
g3-pm8000	15999	79984.0	1.4	0.7	0.6	31993.0	95976.0	10.7	21.8
g2-g22500	44999	179986.7	8.1	12.4	2.9	67494.3	179981.3	5.9	2.0
g2-g62500	124999	499986.0	38.8	62.8	10.3	187494.0	499980.0	24.5	2.4
g3-g8000	15999	79983.3	3.8	5.1	0.8	31992.7	95974.7	11.9	15.1
g3-g64000	127999	639984.0	396.5	786.4	11.6	255993.0	767976.0	857.2	73.6

Table 1. Optimizing over the rooted semimetric polytope

The results in Table 1 clearly show that for these classes of instances the Cost Scaling algorithm actually provide better performances than simplex-type approaches for MCFP, and that all the min-cost flow algorithms outperform the general-purpose LP algorithm. The last column of Table 1 reports the speed factor of CS2 with respect to ILOG-Cplex 8.1 Barrier.

It is known that (5) provides a rather weak upper bound to the max-cut problem. Therefore, the algorithm described here to solve (5) has been designed primarily with the purpose of optimizing over (1), as it will be explained later. Nevertheless, there are instances of max-cut for which one may expect the bound given by (5) to be of some help in practical computation. These are, for example, instances where the weights of the edges incident with a selected node are, in absolute value, sensibly larger than all the other weights. These instances have received a considerable attention in the computational studies of max-cut (or of its equivalent problem, the unconstrained quadratic 0–1 problem) (see, e.g., [4] and [16]). Using our algorithm embedded in a vanilla branch and bound scheme we obtained the results reported in Table 2.

Table 2 reports the result of the branch and bound approach based on the rooted semimetric relaxation on classical instances from the literature [16]. More precisely, instances denoted as “a” and “c” are the ones from [16] generated as in [4] and [18], while instances denoted as “b” are the ones from [16] generated as in [12]. Table 2 reports information about the problem size (n), the density of the graph (d), and the intervals of the random generated weights for the edges

Name	n	d	off-diagonal weights	diagonal weights	nodes	time
1a	50	.1	[-100,100]	[-100,100]	3	0.03
2a	60	.1	[-100,100]	[-100,100]	1	0.03
3a	70	.1	[-100,100]	[-100,100]	159	0.12
4a	50	.2	[-100,100]	[-100,100]	135	0.09
5a	30	.4	[-100,100]	[-100,100]	119	0.06
6a	30	.5	[-100,100]	[-100,100]	75	0.07
7a	100	.0625	[-100,100]	[-100,100]	1	0.03
1b	20	1.	[-100,0]	[0,63]	29	0.03
2b	30	1.	[-100,0]	[0,63]	67	0.05
3b	40	1.	[-100,0]	[0,63]	105	0.11
4b	50	1.	[-100,0]	[0,63]	115	0.20
5b	60	1.	[-100,0]	[0,63]	145	0.42
6b	70	1.	[-100,0]	[0,63]	177	0.87
7b	80	1.	[-100,0]	[0,63]	261	1.81
8b	90	1.	[-100,0]	[0,63]	397	5.12
9b	100	1.	[-100,0]	[0,63]	655	14.57
10b	125	1.	[-100,0]	[0,63]	885	51.53
1c	40	.8	[-50,50]	[-100,100]	2815	1.90
2c	50	.6	[-50,50]	[-100,100]	23275	19.29
3c	60	.4	[-50,50]	[-100,100]	6247	5.82
4c	70	.3	[-50,50]	[-100,100]	8737	8.08
5c	80	.2	[-50,50]	[-100,100]	2191	2.48
6c	90	.1	[-50,50]	[-100,100]	11	0.04
7c	100	.1	[-50,50]	[-100,100]	1	0.03

Table 2. Branch and bound based on the solution of the rooted semimetric relaxation

incident with the selected nodes and for the other edges, respectively. The last two columns of the tables give the outcome in terms of nodes and CPU time of the branch and bound algorithm. The results show that for those problems the rooted semimetric relaxation provides a very good bound allowing their effective solution.

3 Optimizing over the Semimetric Polytope

Let us denote the constraints (4) for one given root node r by $A^r x \leq b^r$. Then, linear optimization over (1) can be written as

$$\max_x \{cx \mid A^r x \leq b^r, \text{ for } r = 1, \dots, n-1\} \quad (7)$$

Note that any two blocks of constraints $A^r x \leq b^r$ and $A^q x \leq b^q$ are in general not disjoint; in fact, only $n-1$ blocks suffice to “cover” all constraints in (1).

We assume that, each time that multiple copies of a constraint (and the corresponding dual multipliers) are present, only one copy is actually considered; accordingly, we will write

$$\min_y \left\{ \sum_{r=1}^{n-1} y^r b^r \mid \sum_{r=1}^{n-1} y^r A^r \geq c \right\} \quad (8)$$

for the dual of (7).

A possible way for solving (7) is to form the Lagrangian relaxation of (7) with respect to all other blocks of constraints but one

$$\max_x \left\{ cx - \sum_{h \neq r} y^h (b^h - A^h x) \mid A^r x \leq b^r \right\} \quad (9)$$

where the y^h , $h \neq r$ are fixed Lagrangian multipliers, and then solve the corresponding Lagrangian Dual

$$\min_y \{v(9) \mid y \geq 0\} \quad (10)$$

where $v(\cdot)$ denotes the optimal objective function value of a problem. This is equivalent to solving problem (8); the Lagrangian multipliers in (10) are precisely the dual variables of (8), except for the $O(n^2)$ dual variables y^r that are *implicitly* handled by the Lagrangian relaxation.

However, some issues arise:

- Problem (10) is a challenging large-scale Non Differentiable Optimization (NDO) problem, therefore it is natural to ask whether such approach can ever compete with a LP-based one?
- How should we choose the root r ? Should we just select r and keep it fixed or a “root hopping” strategy would be preferable?
- Each of the constraint blocks $A^r x \leq b^r$ involves only a “few” ($O(n^2)$) of the “many” ($O(n^3)$) constraints of (7) or, put it in dual terms, the optimization over the single block in (9) can only set a “few” of the “many” variables of (8). Consequently, one wonders if solving (10) has a clear advantage over solving the equivalent Lagrangian Dual

$$\min_{y \geq 0} \left\{ \max_x \left\{ cx - \sum_{r=1}^{n-1} y^r (b^r - A^r x) \mid x \in \{0, 1\}^E \right\} \right\} \quad (11)$$

of (7) with respect to *all* blocks of constraints?

- Which NDO algorithm has to be used for solving the Lagrangian Duals?
- Are there alternative Lagrangian approaches based on reformulations of (7), such as Lagrangian Decomposition, that provide more convenient ways for solving (8)?

We experimentally evaluated the proposed approaches with a large-scale comparison. Due to the large number of constraints, each algorithm was embedded into a cutting plane scheme, where first a formulation with no explicit constraint is taken and then it is iteratively enlarged by adding explicit constraints violated by the current primal solution.

The test-bed was made by 175 instances of the same type of graphs described above, but in this context sparse graphs were “completed” by adding zero-cost edges, so that the separation procedure was done by trivial enumeration of all triangle inequalities and the results were not affected by the degree of sophistication of the algorithm used to separate the cycle inequalities.

Our experiments showed that these large-scale NDO problems, at least for the instances we tested, appear to be relatively easy to solve, even if a fixed root r is chosen with a simple heuristic. This is shown in Table 3, where the results obtained by the approach of [3] for solving (10) and (11) (columns “V1” and “V0”), those of a corresponding bundle-type approach (columns “B1” and “B0”) and those of a finely tuned code using the state-of-the-art general-purpose LP solver ILOG-Cplex 8.1 (column “ILOG-Cplex”) are compared. Computing times are in seconds on an Athlon MP 2400+.

The table clearly shows that the Lagrangian approaches, in particular the bundle-based one using (5) as subproblem (B1), provides solutions with very low primal and dual relative gaps (columns “PGap” and “DGap”, an empty entry in those columns corresponding to a gap not larger than $1e-10$) in a small fraction of the time required by the LP solver.

The table also shows that “x1” approaches are in general much more efficient than “x0” ones, with the notable exception of complete (clique) graphs. As far as the “Vx” versus “Bx” comparison is concerned, most often the bundle method obtains much better dual precision in comparable or even less time; furthermore, the subgradient approach never produces primal solutions of even moderately good quality, whereas the bundle approach always produces solutions of acceptable — and most often of excellent — quality. However, especially for the largest instances the subgradient can be significantly faster.

We remark that alternative, more complex Lagrangian approaches — that we do not describe here for space reasons — have been tested, but they have not been found to be competitive with those presented here.

The above analysis allows us to conclude that Lagrangian approaches for linear optimization over (1) are competitive with LP-based. On all “structured” instances, exploiting the efficient algorithms for (5) is instrumental. If accurate primal solutions are required a bundle approach is also instrumental; the bundle approach is also necessary for obtaining accurate dual solutions in several cases, and either competitive or downright faster in many others. However, on some large-scale instances, or if only a rough dual bound has to be obtained quickly, the subgradient algorithm may provide an interesting alternative.

	ILOG-Cplex						B1					
	V0			V1			B0			B1		
	DGap	Pgap	time	DGap	Pgap	time	DGap	Pgap	time	DGap	Pgap	time
c25	0.28	2e-7	8e-3	0.04	1e-7	6e-3	0.47	1e-8	0.39	7e-9	7e-6	0.27
c50	8.97	8e-4	7e-3	0.54	6e-4	5e-3	2.21	7e-9	1e-5	4.00	4e-6	4.80
c75	69.03	6e-4	4e-3	1.87	4e-4	3e-3	6.80	8e-9	6e-6	13.33	4e-9	19.38
c100	386.60	5e-4	5e-3	4.11	4e-4	5e-3	13.88	8e-9	9e-6	34.80	3e-9	42.79
c125	1409.20	5e-4	7e-3	8.54	3e-4	3e-3	26.43	2e-9	1e-5	86.45	1e-9	1e-5
c150	3729.51	5e-4	5e-3	14.90	3e-4	3e-3	50.35	4e-9	1e-5	138.03	1e-9	2e-5
p50	13.80	5e-8	8e-3	0.50	3e-3	0.99		7e-9	4.11			0.67
p100	762.15	4e-6	3e-2	5.58	2e-2	6.20		1e-7	1557.56			5.04
p150	8877.42	1e-4	5e-2	24.97	2e-8	5e-2	21.53	2e-9	5e-6	5448.50		19.78
s21	0.08	3e-3	0.03		5e-4	0.12			0.05			0.02
s56	25.96	1e-2	0.67	3e-9	2e-2	2.14		4.09				1.46
s91	462.03	3e-7	2e-2	4.07	2e-2	6.38		29.05				5.17
s136	5492.87	3e-5	3e-2	15.40	1e-1	19.75		2e-9	3577.35			31.34
g2-pm25	0.28	4e-4	2e-3	0.06	2e-7	2e-3	0.31	2e-9	0.42			0.08
g2-pm49	15.01	2e-4	8e-3	0.61	1e-8	6e-3	1.71	7e-9	4e-7	7.31		1.21
g2-pm81	233.91	7e-5	2e-2	2.92	7e-8	1e-2	5.69	7e-9	2e-6	6240.71		9.05
g2pm-100	1218.55	1e-3	5e-2	13.64	5e-6	5e-2	13.06	6e-9	2e-5	8960.96		11.71
g2pm-144	9436.36	1e-3	6e-2	23.03	1e-4	9e-2	50.02	3e-6	5e-4	11024.20	6e-9	141.05
g3-pm27	0.70	6e-4	2e-3	0.07	9e-8	4e-3	0.57	1e-9	3e-8	4.71		0.97
g3-pm64	59.23	7e-5	2e-2	1.66	4e-2	5.22		5e-9	3e-7	132.73		2.28
g3-pm125	3427.77	1e-3	5e-2	16.88	2e-5	9e-2	52.03	5e-8	4e-6	6734.21		32.99
g2-g25	0.28	3e-3	0.05		1e-3	0.15			0.10			0.05
g2-g49	13.98	7e-3	0.47		5e-3	0.98			1.66			0.45
g2-g81	187.43	1e-7	3e-2	2.77		2e-2	4.44		15.11			2.99
g2-g100	788.55	6e-6	5e-2	6.02		9e-2	6.93		90.01			7.37
g2-g144	9050.67	7e-5	7e-2	23.81	2e-9	1e-1	29.36		2e-9	1528.10	3e-8	60.84
g3-g27	0.49	4e-3	0.06		5e-4	0.26			0.21			0.07
g3-g64	58.45	3e-8	2e-2	1.39		2e-2	3.63			10.24		1.98
g3-g125	3564.98	4e-5	6e-2	16.20	1e-9	1e-1	39.02			288.21		26.85

Table 3. Main table of results

References

1. L. BAHENSE, N. MACULAN, AND C. SAGASTIZÁBAL, *The volume algorithm revisited: relation with bundle methods*, Mathematical Programming, 94 (2002), pp. 41–70.
2. BARAHONA, F., *Network Design Using Cut Inequalities*, SIAM Journal on Optimization, 6 (1996), pp. 823–837.
3. F. BARAHONA AND R. ANBIL, *The volume algorithm: Producing primal solutions with a subgradient method*, Mathematical Programming, 87 (2000), pp. 385–400.
4. F. BARAHONA, M. JÜNGER, AND G. REINELT, *Experiments in quadratic 0–1 programming*, Mathematical Programming, 44 (1989), pp. 127–137.
5. F. BARAHONA AND A. MAHJOUB, *On the cut polytope*, Mathematical Programming, 36 (1986), pp. 157–173.
6. BIENSTOCK, D., CHOPRA, S., GÜNLÜK, O., AND TSAI, C., *Minimum Cost Capacity Installation for Multicommodity Network Flows*, Mathematical Programming, 81 (1998), pp. 177–199.
7. M. DEZA AND M. LAURENT, *Geometry of Cuts and Metrics*, vol. 15 of Algorithms and Combinatorics, Springer-Verlag, Berlin, 1997.
8. A. FRANGIONI, *Generalized bundle methods*, SIAM Journal on Optimization, 13 (2002), pp. 117–156.
9. FRANGIONI, A. AND MANCA, A., *A Computational Study of Cost Reoptimization for Min Cost Flow Problems*, INFORMS Journal on Computing, to appear.
10. D. FULKERSON, A. HOFFMAN, AND M. MCANDREW, *Some properties of graphs with multiple edges*, Canadian Journal of Mathematics, 17 (1965), pp. 166–177.
11. A. GOLDBERG, *An efficient implementation of a scaling minimum-cost flow algorithm*, Journal of Algorithms, 22 (1997), pp. 1–29.
12. V. GULATI, S. GUPTA, AND A. MITTAL, *Unconstrained quadratica bivalent programming problem*, European Journal of Operational Research, 15 (1984), pp. 121–125.
13. D. HOCHBAUM, *Monotonizing matrices with up to two nonzeroes per column*, Operations Research Letters, 32 (2003), pp. 49–58.
14. A. LÖBEL, *Solving large-scale real-world minimum-cost flow problems by a network simplex method*, Technical Report SC-96-7, ZIB Berlin, 1996.
15. M. LOMONOSOV, *Combinatorial approaches to multiflow problems*, Discrete Applied Mathematics, 11 (1985), pp. 1–93.
16. P. PARDALOS AND G. RODGERS, *Computational aspects of a branch and bound algorithm for quadratic zero-one programming*, Computing, 45 (1990), pp. 131–144.
17. V. VAZIRANI AND M. YANNAKAKIS, *Suboptimal cuts: Their enumeration, weight and number*, in Proceedings of the 19th International Colloquium on Automata, Languages, and Programming, W. Kuich, ed., vol. 623 of Lecture Notes in Computer Science, New York, NY, 1992, Springer-Verlag, pp. 366–377.
18. A. WILLIAMS, *Quadratic 0–1 programming using the roof dual with computational results*, Tech. Report RUTCOR-8-85, State University of New Jersey, 1985.

Appendix

Sketch of the proof of Theorem 3. With a couple of simple variable transformations it is possible to rewrite system (5) as

$$\max d \cdot z$$

subject to

$$\begin{aligned} z_{ri} + z_{rj} - z_{ij} &\leq 1, & ij \in E_+^r \\ -z_{rh} + z_{rk} - z_{ij} &\leq 0, & ij \in E_-^r \text{ and } \begin{cases} h = \min\{i, j\} \\ k = \max\{i, j\} \end{cases} \\ z_{ri} &\leq 1, & i \in V^r \\ z &\geq 0, \end{aligned} \tag{12}$$

where d_{ri} , for $i \in V^r$, coincides with d_i as defined in (6), and $d_{ij} := 2|c_{ij}|$, for $ij \in E^r$. Then, it is not difficult to see that the dual of (12) can be interpreted as a network flow problem on a mixed network in which there are undirected edges (associated with constraints corresponding to $ij \in E_+^r$), and directed arcs (associated with constraints corresponding to $ij \in E_-^r$), plus directed arcs from/to the root node r to/from all other nodes $i \in V^r$. A simple example of a complete graph of four nodes is shown in Figure 1, while the corresponding mixed network is given in Figure 2. Specifically, in Figure 2 node labels in boldface and in italic-

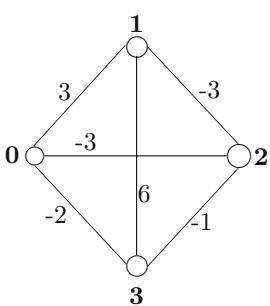


Fig. 1. Weighted undirected graph of four nodes

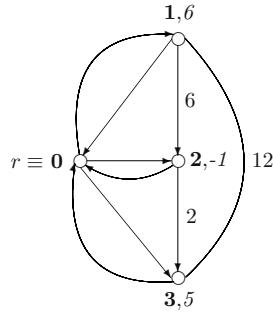


Fig. 2. Network interpretation of the dual of (12)

ics denotes node identifiers and node supply/demand, respectively; edge labels denote edge capacities. Directed arcs from node r to the other nodes have cost 1, as well as the edges of the mixed network, while the directed arcs between nodes $i, j \in V^r$ have cost 0. Finally, arcs from/to node r have infinite capacity.

Obviously, in the special case in which $E_+^r = \emptyset$, the mixed network becomes a directed network and the dual of (12) is already a min-cost flow problem.

When $E_+^r \neq \emptyset$, to overcome the difficulty associated with the presence of undirected edges in the network representing the dual of system (12) we make

use of a transformation proposed in [10] and recently pointed out in [13]. The transformation works as follows: (a) for each node $i \in V^r$ we make a replica, say i' , connected to node r with two arcs ri' and $i'r$; (b) for each arc ij in the mixed network we have an additional arc $j'i'$; and (c) each edge ij in the mixed network is replaced by two arcs $i'j$ and $j'i$. Costs and capacities of the arcs in the directed network are naturally inherited from edges and arcs in the mixed network, while the supply/demand associated with each duplicated node is the opposite of the original one. The directed graph in the special case of the example of figures 1 and 2 is shown in Figure 3.

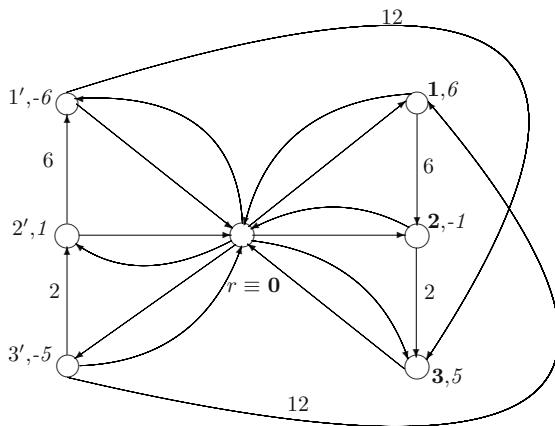


Fig. 3. The directed network corresponding to the mixed network of Figure 2

Author Index

- Apollonio, N., 388
Asgeirsson, E., 116
Avella, P., 16

Berry, J., 116
Bertsimas, D., 86
Bessy, S., 132
Boros, E., 152

Chen, B., 219
Codato, G., 178
Correa, J.R., 59, 283

Dash, S., 33

Elbassioni, K., 152

Farias, I.R. de, Jr., 163
Fischetti, M., 178
Frangioni, A., 431
Fukasawa, R., 1

Günlük, O., 33
Gurvich, V., 152

Haws, D., 244
Hemmecke, R., 244
Huggins, P., 244

Iwata, S., 352

Kaibel, V., 401
Karzanov, A.V., 368
Khachiyan, L., 152
Király, T., 416

Lang, K., 325
Lee, J., 271
Letchford, A.N., 196
Levi, R., 206
Levin, A., 298
Lodi, A., 431
Loera, J.A. De, 244, 338
Lozovanu, D., 74
Lysgaard, J., 1

Magnanti, T.L., 234
Makai, M., 416

Margot, F., 271
Mattia, S., 16
Moriguchi, S., 352
Murota, K., 352

Onn, S., 338

Pap, G., 139
Perakis, G., 46
Phillips, C.A., 116
Phillips, D.J., 116
Poggi de Aragão, M., 1

Rao, S., 325
Ravi, R., 101
Reinelt, G., 196
Reis, M., 1
Rinaldi, G., 431

Sassano, A., 16
Schulz, A.S., 59, 283
Sebő, A., 256, 388
Shmoys, D.B., 206
Sinha, A., 101
Stein, C., 116
Stier Moses, N.E., 59
Stratila, D., 234
Swamy, C., 206
Szegő, L., 256

Theis, D.O., 196
Thiele, A., 86
Thomassé, S., 132

Uchoa, E., 1

Vygen, J., 308

Wein, J., 116
Werneck, R.F., 1
Woeginger, G.J., 298

Ye, Y., 219
Yoshida, R., 244

Zhang, J., 219