

Boris I. Goldengorin  
Valery A. Kalyagin  
Panos M. Pardalos *Editors*

# Models, Algorithms, and Technologies for Network Analysis

Proceedings of the Second International  
Conference on Network Analysis

# **Springer Proceedings in Mathematics & Statistics**

---

Volume 59

---

For further volumes:  
[www.springer.com/series/10533](http://www.springer.com/series/10533)

# **Springer Proceedings in Mathematics & Statistics**

---

---

This book series features volumes composed of selected contributions from workshops and conferences in all areas of current research in mathematics and statistics, including OR and optimization. In addition to an overall evaluation of the interest, scientific quality, and timeliness of each proposal at the hands of the publisher, individual contributions are all refereed to the high quality standards of leading journals in the field. Thus, this series provides the research community with well-edited, authoritative reports on developments in the most exciting areas of mathematical and statistical research today.

Boris I. Goldengorin • Valery A. Kalyagin •  
Panos M. Pardalos

Editors

# Models, Algorithms, and Technologies for Network Analysis

Proceedings of the Second International  
Conference on Network Analysis

 Springer

*Editors*

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

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

Valery A. Kalyagin  
National Research University  
Higher School of Economics  
Nizhny Novgorod, Russia

ISSN 2194-1009

ISSN 2194-1017 (electronic)

Springer Proceedings in Mathematics & Statistics

ISBN 978-1-4614-8587-2

ISBN 978-1-4614-8588-9 (eBook)

DOI 10.1007/978-1-4614-8588-9

Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2013949859

Mathematics Subject Classification (2010): 90-02, 90C31, 90C27, 90C09, 90C10, 90C11, 49L20, 90C35, 90B06, 90B10, 90B15, 90B18, 90B40, 90B80, 68R10

© Springer Science+Business Media New York 2013

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

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

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

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

*The nature of God is a circle  
of which the center is everywhere  
and the circumference is nowhere.*

*Empedocles (Greek philosopher,  
c. 490–430 BC)*

# Preface

This volume contains two types of papers either presented on the 2nd International Conference on Network Analysis (which took place on May 7–9, 2012 in Nizhny Novgorod, Russia) or submitted within an open call for papers reflecting the activities of LATNA—Laboratory of Algorithms and Technologies for Networks Analysis at the Higher School of Economics. This conference is sponsored by LATNA. All participants, authors and editors are gratefully acknowledge the financial support by The Russian Federation Government Grant, ag.11.G34.31.0057. Our special thanks going to the LATNA's staff, especially to Dr. Mikhail Batsyn for many efforts in the process of collecting and reviewing the submitted papers including the compilation of this final volume. We are grateful to the members of Program Committee and the external referees for their careful reading and many useful comments essentially improving the quality of this book. The success of this conference was pre-defined by the following distinguished plenary speakers:

1. Ding-Zhu Du (University of Texas at Dallas, USA) Min-Weight Connected Sensor Cover and Max-Lifetime Target Coverage
2. Christodoulos Floudas (Princeton University, USA) Towards Large Scale Deterministic Global Optimization
3. Boris Mirkin (Higher School of Economics, Russia) Representing Activities by Taxonomy Concepts: Clustering and Lifting
4. Mauricio Resende (AT&T, USA) Randomized Algorithms for the Handover Minimization Problem in Wireless Network Design

The plenary lectures are unforgettable from many points of view. They were not only based on the state-of-the art in each topic represented by the corresponding speaker but have shown how to organize an overview lecture in an easy understandable way bringing even undergraduate students to the frontier of science in networks and algorithms. This volume contains many new results in modeling and powerful algorithmic solutions applied to the problems in vehicle routing, single machine scheduling, modern financial markets, cell formation in group technology, comparison of brain activities of left- and right-handers, speeding up algorithms for the maximum clique problem, analysis, and applications of different measures in clustering.

There are three overview papers in this volume: an overview of Kernel Principal Component Analysis including its implementation of the improved nearest neighbor and kernel regression methods in MATLAB, an overview of clique relaxation models and their applications, and an overview of the double partition technique together with research progress on approximations for the minimum sensor cover problem. Also this volume contains a new formulation of the facility layout problem and its applications to different issues in health care as well as graph-based analysis applied to the BRIC countries stock markets.

More than 60 researchers including undergraduate students from universities, institutes, governmental agencies, and industrial companies worldwide attended the conference.

Gainesville, FL, USA  
Nizhny Novgorod, Russia  
Gainesville, FL, USA

Boris I. Goldengorin  
Valery A. Kalyagin  
Panos M. Pardalos



# Program of the Second International Conference on Network Analysis 2012

## Monday, May 7th, 2012

Room 313 HSE, 25/12 Bolshaya Pecherskaya Str.

15:00–15:30 Panos M. Pardalos  
*Second International Conference on Network Analysis 2012*

15:30–16:20 Christodoulos A. Floudas  
*Towards Large Scale Deterministic Global Optimization*

16:20–16:40 Coffee Break

16:40–18:10 Session 1

Ludmila Egorova  
*Behavioral model of stock exchange*

Dmitry Malyshev  
*On expanding operators for the independent set problem*

Dmitry Mokeev  
*Structural and complexial properties of  $P_3$ -könig graphs*

Victor Zamaraev  
*A heuristics for the weighted independent set problem*

## Tuesday, May 8th, 2012

Room 313 HSE, 25/12 Bolshaya Pecherskaya Str.

10:00–10:50 Boris Mirkin  
*Representing activities by taxonomy concepts: clustering and lifting*

10:50–11:10 Coffee Break

11:10–12:30 Session 1

Pando G. Georgiev

*Innovative tools for analyzing state transitions and evolution of complex dynamic networks*

Alexey Yashunsky

*Using online social networks for social geography studies*

Alexander Rubchinsky

*A new algorithm of network decomposition and its application for stock market analysis*

12:30–14:00 Lunch Break

14:00–14:50 Ding-Zhu Du

*Min-weight connected sensor cover and max-lifetime target coverage*

14:50–15:50 Session 2

Anton Kocheturov

*Market graph analysis by means of the  $p$ -median problem*

Mikhail Batsyn

*Applying tolerances to the asymmetric capacitated vehicle routing problem*

Evgeny Maslov

*Complex approach to solving the maximum clique problem*

15:50–16:10 Coffee Break

16:10–17:30 Session 3

Grigory Bautin

*Markov chains in modeling of the Russian financial market*

Dmitry Gorbunov

*Simulation of pedestrian crowds with anticipation using cellular automata approach*

Pankaj Kumar

*Behavioural dynamics in stock market*

Lazarev Evgeny Alexandrovich

*Bi-criteria model and algorithms of solving data transmission network optimization problem*

## **Wednesday, May 9th, 2012**

Room 313 HSE, 25/12 Bolshaya Pecherskaya Str.

9:30–10:20 Mauricio G. C. Resende

*Randomized algorithms for the handover minimization problem in wireless network design*

10:20–10:40 Coffee Break

10:40–12:20 Session 1

D.V. Kasatkin

*Synaptic cellular automaton for description the sequential dynamics of excitatory neural networks*

Pavel Sukhov

*Heuristic algorithm for the single machine scheduling problem*

Ilya Bychkov

*“Patterns” for solving the cell formation problem*

Peter Koldanov

*Statistical properties of the market graph*

# Contents

<b>Tolerance-Based vs. Cost-Based Branching for the Asymmetric Capacitated Vehicle Routing Problem . . . . .</b>	<b>1</b>
Mikhail Batsyn, Boris Goldengorin, Anton Kocheturov, and Panos M. Pardalos	
<b>Lower and Upper Bounds for the Preemptive Single Machine Scheduling Problem with Equal Processing Times . . . . .</b>	<b>11</b>
Mikhail Batsyn, Boris Goldengorin, Pavel Sukhov, and Panos M. Pardalos	
<b>Comparative Analysis of Two Similarity Measures for the Market Graph Construction . . . . .</b>	<b>29</b>
Grigory A. Bautin, Valery A. Kalyagin, and Alexander P. Koldanov	
<b>Heuristic Algorithm for the Cell Formation Problem . . . . .</b>	<b>43</b>
Ilya Bychkov, Mikhail Batsyn, Pavel Sukhov, and Panos M. Pardalos	
<b>Efficiency Analysis of Branch Network . . . . .</b>	<b>71</b>
Petr A. Koldanov	
<b>EEG Coherence in Right- and Left-Handers in Passive Visual Perception of Lines with Different Slope Angles . . . . .</b>	<b>85</b>
Maxim Viktorovich Lukoyanov	
<b>Speeding up MCS Algorithm for the Maximum Clique Problem with ILS Heuristic and Other Enhancements . . . . .</b>	<b>93</b>
Evgeny Maslov, Mikhail Batsyn, and Panos M. Pardalos	
<b>Summary and Semi-average Similarity Criteria for Individual Clusters</b>	<b>101</b>
Boris Mirkin	
<b>Kernel Principal Component Analysis: Applications, Implementation and Comparison . . . . .</b>	<b>127</b>
Daniel Olsson, Pando Georgiev, and Panos M. Pardalos	

<b>Distance-Based Clique Relaxations in Networks: <math>s</math>-Clique and <math>s</math>-Club</b> . . . . .	<b>149</b>
Shahram Shahinpour and Sergiy Butenko	
<b>GRASP with Path-Relinking for Facility Layout</b> . . . . .	<b>175</b>
R.M.A. Silva, M.G.C. Resende, P.M. Pardalos, G.R. Mateus, and G. De Tomi	
<b>Comparative Analysis of the BRIC Countries Stock Markets Using Network Approach</b> . . . . .	<b>191</b>
Arsenii Vizgunov, Andrey Glotov, and Panos M. Pardalos	
<b>Sensor Cover and Double Partition</b> . . . . .	<b>203</b>
Lidong Wu, Weili Wu, Zaixin Lu, Yuqing Zhu, and Ding-Zhu Du	

# Tolerance-Based vs. Cost-Based Branching for the Asymmetric Capacitated Vehicle Routing Problem

Mikhail Batsyn, Boris Goldengorin, Anton Kocheturov,  
and Panos M. Pardalos

**Abstract** In this chapter, we consider the asymmetric capacitated vehicle routing problem (ACVRP). We compare the search tree size and computational time for the bottleneck tolerance-based and cost-based branching rules within a branch-and-bound algorithm on the FTV benchmark instances. Our computational experiments show that the tolerance-based branching rule reduces the search tree size by 45 times and the CPU time by 2.8 times in average.

**Keywords** Vehicle routing · Branch-and-bound · Tolerance · Branching strategy

## 1 Introduction

The vehicle routing problem (VRP) consists in finding  $K$  disjoint cyclic routes (one route for each of  $K$  vehicles) in a complete weighted graph starting and ending at the depot (vertex 0) and covering  $n$  clients (vertices  $1, \dots, n$ ) such that the total traveled distance (the total weight of all  $K$  routes) is minimized. In the capacitated vehicle routing problem (CVRP) every client  $i$  has a certain demand  $d_i$  of goods, which should be delivered from the single depot, and every vehicle has a limited capacity  $C$  of goods it can supply to clients. The CVRP is one of the difficult combinatorial

---

M. Batsyn (✉) · A. Kocheturov · P.M. Pardalos

Laboratory of Algorithms and Technologies for Networks Analysis, National Research University  
Higher School of Economics, 136 Rodionova, Nizhny Novgorod, Russian Federation  
e-mail: [mbatsyn@hse.ru](mailto:mbatsyn@hse.ru)

A. Kocheturov

e-mail: [antrubler@gmail.com](mailto:antrubler@gmail.com)

P.M. Pardalos

e-mail: [pardalos@ufl.edu](mailto:pardalos@ufl.edu)

B. Goldengorin · P.M. Pardalos

Center of Applied Optimization, University of Florida, 401 Weil Hall, P.O. Box 116595,  
Gainesville, FL 32611-6595, USA

B. Goldengorin

e-mail: [goldengorin@ufl.edu](mailto:goldengorin@ufl.edu)

optimization problems (see, e.g., Toth and Vigo (2002) [15] and Golden et al. (2008) [9]). The largest instances which can be solved by the state-of-art algorithms have only 135 vertices (Lysgaard et al. (2004) [13], Baldacci et al. (2004) [1], Fukasawa et al. (2006) [6], Baldacci et al. (2008) [2], Baldacci et al. (2011) [3]).

The majority of the papers are devoted to the symmetric CVRP. Good overviews of the recent results for the CVRP could be found in papers of Laporte (2009) [12] and Baldacci et al. (2012) [4]. A complete and detailed description of different types of the VRP and algorithms for solving VRP problems is provided in the book of Toth and Vigo (2002) [15]. The asymmetric CVRP (ACVRP) has received less attention in recent decades. After the paper of Fischetti et al. (1994) [5], there are almost no exact algorithms for the ACVRP except maybe Pessoa et al. (2008) [14]. According to the results reported by Pessoa et al. (2008) [14], branch-and-bound algorithms are more successful in solving ACVRP problems with a small number of vehicles, while branch-and-cut-and-price approach is better for a greater number of vehicles. The best results for the ACVRP with 2–3 vehicles were obtained in Fischetti et al. (1994) [5].

In this chapter, we compare two branching rules within a branch-and-bound algorithm for the ACVRP. They are the tolerance-based branching rule, which uses the bottleneck upper tolerance and the classical cost-based branching rule. Our tolerance-based branching strategy is similar to the approach suggested by Golden-gorin et al. (2004) [10] for the ATSP problem. Compared to the cost-based branching rule, our tolerance-based branching rule allows to reduce the search tree size considerably. The notion of tolerance is well known in sensitivity analysis for linear programming problems (see, e.g., Gal and Greenberg, 1997 [7]). A tolerance is a maximal possible change of a parameter value (an arc weight in the case of the ACVRP) for which the current optimal solution remains optimal. A tolerance-based approach was suggested by Goldengorin et al. (2004) [10] and proved its efficiency for the asymmetric traveling salesman problem (ATSP). In this paper, we present some preliminary computational results showing a comparison of the classical cost-based branching rule and the suggested tolerance-based one. Our computational experiments for seven FTV benchmark instances show that the search tree size is reduced by 45 times in average and the computational time is reduced by 2.8 times in average.

The chapter is organized as follows. In the next section, we give the formulation of the ACVRP. The suggested branch-and-bound algorithm is described in Sect. 3. The computational results are presented in Sect. 4. Section 5 concludes the chapter with a short summary.

## 2 The Asymmetric Capacitated Vehicle Routing Problem

The ACVRP has several mathematical programming models. Below we provide the two-index vehicle flow formulation (Toth and Vigo, 2002 [15]). The ACVRP is described by a directed weighted graph  $G = (V, A)$ , where  $V = \{0, 1, \dots, n\}$  is a

set of vertices (vertex 0 is a depot, vertices  $1, \dots, n$  are clients) and  $A$  is a set of arcs. Arc weights are given by the cost matrix  $c_{ij}$ . There are  $K$  vehicles of identical capacity  $C$  located at the depot. A given quantity of goods  $d_i$  should be delivered to client  $i$ . The objective is to find  $K$  disjoint cyclic routes from the single depot, one route for each vehicle, serving all the  $n$  clients with their demands  $d_i$  such that the total traveling cost is minimized.

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \quad (1)$$

subject to

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \setminus \{0\} \quad (2)$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V \setminus \{0\} \quad (3)$$

$$\sum_{i \in V} x_{i0} = K \quad (4)$$

$$\sum_{j \in V} x_{0j} = K \quad (5)$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq \left\lceil \sum_{i \in S} d_i / C \right\rceil \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset \quad (6)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V \quad (7)$$

Variable  $x_{ij}$  is equal to 1 when arc  $(i, j)$  belongs to the optimal solution and is equal to 0 otherwise. The objective function (1) requires that the total weight of the arcs in the solution is minimized. Constraints (2) and (3) require that only one arc in the solution goes to a client and only one arc leaving this client. Constraints (4) and (5) require that there are  $K$  incoming and  $K$  outgoing arcs in the depot. Capacity-cut constraints (6) require that any set  $S$  of clients with total demand  $\sum_{i \in S} d_i$  should be served by the number of vehicles with capacity  $C$  which is enough to cover such a demand.

### 3 Algorithm Description

In our algorithm, we use the assignment problem (AP) as a relaxation of the ACVRP. For this purpose, row  $c_{0j}$  and column  $c_{i0}$  of the cost matrix  $c_{ij}$  are duplicated  $K$  times. This means that we make  $K$  copies of the depot vertex in order to transform constraints (4) and (5) into standard AP constraints. The AP solution is represented by a set of disjoint cycles some of which could be infeasible for the ACVRP problem. There are two types of infeasible cycles: a cycle without the depot, and an



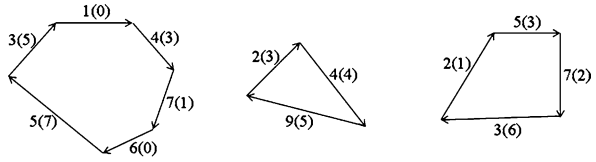
overloaded cycle with the total demand of clients greater than a vehicle capacity. In order to reduce the search tree size, we also consider as infeasible too small cycles with the total demand of clients smaller than  $\sum_{i \in V} d_i - (K - 1)C$ . In the case of such a cycle, the remaining  $(K - 1)$  vehicles cannot serve the remaining clients because the total demand of these clients is greater than  $(K - 1)C$ .

In each node of the solution tree, we solve the assignment problem. If the AP solution is feasible for the ACVRP, then the current branch is completed. Otherwise all the infeasible cycles are considered and one of them is destroyed by means of removing one of its arcs. To remove arc  $a$ , its traveling cost  $c_a$  is replaced with  $\infty$ . For example, if there is an infeasible cycle of three arcs  $x, y, z$  in the AP solution, then in the first branch of the solution tree we remove arc  $x$  (branch  $x\bar{x}$ ). In the second branch, this arc is forcibly included to the AP solution (by setting  $c_x = -\infty$ ), and arc  $y$  is removed (branch  $x\bar{y}$ ). Finally, in the third branch arcs  $x$  and  $y$  are forcibly included to the AP solution, and arc  $z$  is removed (branch  $xy\bar{z}$ ). The question is which cycle is better to destroy and what arc to remove first.

We suggest to use upper tolerances for efficient selecting of a branch in a branch-and-bound algorithm. Upper tolerance  $u_a$  of arc  $a$  from the optimal AP solution is the maximal value on which the cost of this arc  $c_a$  can be increased so that this arc still belongs to the optimal solution. It is not difficult to prove that an upper tolerance can be computed using the following formula:  $u_a = f_{-a} - f$ , where  $f$  is the value of the AP objective function and  $f_{-a}$  is the value of this function for the AP in which arc  $a$  is removed (Goldengorin et al., 2004 [10]). Thus, upper tolerance  $u_a$  also shows how much the objective function is increased when we remove arc  $a$ .

We calculate upper tolerances of arcs in the infeasible cycles and first remove the arc which has the minimal tolerance in its cycle. This causes the minimal possible increase of the objective function (the total traveling cost), and thus we cannot miss the branch with the optimal solution of the original ACVRP problem. We call this minimal tolerance in a cycle a “cycle tolerance”, because it shows the increase in the objective function which we cannot avoid when destroying this cycle. Since all the infeasible cycles cannot be present in the optimal solution of the original problem, then we cannot avoid the increase in the objective function equal to the maximal of the tolerances of infeasible cycles. We call this tolerance a “bottleneck tolerance”—the maximal tolerance among the minimal tolerances of arcs of each infeasible cycle. In order to find the optimal value of the original problem, it is necessary to destroy the cycle with the bottleneck tolerance because this tolerance gives the maximal increase in the objective function without jumping over its unknown optimal value for the ACVRP. A classical cost-based branching strategy selects the smallest cycle to destroy (with the smallest number of arcs) and first removes the longest arc (the arc with the greatest weight).

Let the three cycles shown in Fig. 1 be infeasible for some ACVRP. Numbers  $c_a(u_a)$  near every arc show its weight  $c_a$  and upper tolerance  $u_a$ . The minimal tolerances of cycles are equal to 0, 3, 1, correspondingly. According to the tolerance-based branching strategy, the second cycle should be destroyed and its arc with weight 2 and tolerance 3 (which is a bottleneck tolerance) should be removed first. The cost-based approach also destroys the second cycle but removes the longest arc with weight 5 first.

**Fig. 1** Infeasible cycles

In all the procedures of our algorithm described below, the following variables are used:

- $V$ —the list of vertices in the graph
- $A$ —the list of arcs in the graph
- $x$ —the solution of the AP,  $x_{ij} = 1$  if vertex  $j$  is assigned to follow vertex  $i$  in a cycle
- $c$ —the cost matrix
- $c_a$ —the cost (weight) of arc  $a$
- $c_a^0$ —the original cost of arc  $a$
- $u_a$ —the upper tolerance of arc  $a$
- $f$ —the total cost for the current AP solution
- $f_{-a}$ —the total cost for the AP solution after removing arc  $a$
- $f^*$ —the total cost for the best feasible ACVRP solution found so far
- $r$ —the current cyclic route (cycle) in the current AP solution
- $R$ —the set of cyclic routes (cycles) representing the current AP solution
- $|r|$ —the number of arcs in cycle  $r$
- $K$ —the number of vehicles
- $C$ —the capacity of each vehicle
- $d_i$ —the demand of client  $i$
- $d_r$ —the total demand of clients in the current cycle
- $d_{\min}$ —the minimal total demand of a feasible cycle
- $a^*$ —the arc with the bottleneck tolerance
- $r^*$ —the cycle with the arc having the bottleneck tolerance
- $u^*$ —the bottleneck tolerance
- $u_{\min}$ —the minimal tolerance in the current cycle
- $a_{\min}$ —the arc with the minimal tolerance in the current cycle

The main procedure of our algorithm is a recursive function **BRANCH-AND-BOUND**( $c$ ) (Algorithm 1). First, we solve a relaxed problem which is the assignment problem (AP) in our approach. We apply the JV algorithm by Jonker and Volgenant (1987) [11] for this purpose (function **SOLVE-ASSIGNMENT-PROBLEM**( $c$ ,  $R$ ), Algorithm 2). A solution of the AP is a set of disjoint cycles covering all the vertices. The value  $f$  of the AP objective function gives a lower bound for the optimal value of the ACVRP objective function on the current branch of the search tree. That is why if this value is not less than the value  $f^*$  of the best feasible solution found so far, then this branch is completed.

Otherwise, we check if this solution is feasible for the ACVRP (function **IS-FEASIBLE**( $R$ ), Algorithm 3). If it is the case, this solution replaces the currently

**Algorithm 1** Main recursive procedure

---

```

function BRANCH-AND-BOUND( $c$ )
   $f = \text{SOLVE-ASSIGNMENT-PROBLEM}(c, R)$ 
  if  $f > f^*$  then
    return
  end if
  if IS-FEASIBLE( $R$ ) then
     $f^* = f$ 
    return
  end if
  COMPUTE-TOLERANCES( $c, f, R, a, r, u$ )
  while  $f + u_a < f^*$  do
     $c_a^0 = c_a$  ▷ backup the original cost of arc  $a$ 
     $c_a = +\infty$  ▷ remove arc  $a$  from the solution
    BRANCH-AND-BOUND( $c$ )
     $c_a = -\infty$  ▷ include arc  $a$  to the solution
     $u_a = +\infty$  ▷ remove this tolerance from consideration
     $a = \arg \min_{a \in r} (u_a)$  ▷ find next minimal tolerance
  end while
  for  $a \in \{a \mid c_a = -\infty\}$  do ▷ restore the original arc costs
     $c_a = c_a^0$ 
  end for
end function

```

---

**Algorithm 2** Runs the JV algorithm to solve the AP

---

```

function SOLVE-ASSIGNMENT-PROBLEM( $c, R$ )
   $f = \text{JV}(c, x)$  ▷ JV algorithm by Jonker and Volgenant (1987) [11]
  Get cycles  $R$  from assignments  $x$ 
  return  $f$ 
end function

```

---

best solution and the branch is completed. To check the feasibility, we check every cycle  $r$  to have the depot (vertex 0) and the total demand of clients  $d_r$  not less than the minimal possible demand for one cycle  $d_{\min}$  and not greater than the vehicle capacity  $C$ . The minimal cycle demand is calculated by the following formula:  $d_{\min} = \sum_{i \in V} d_i - (K - 1)C$ . If the cycle total demand is less than  $d_{\min}$  then the remaining  $(K - 1)$  vehicles capacity is not enough to cover the demand of the remaining clients.

If the AP solution has infeasible cycles, we calculate the upper tolerances  $u$  of arcs in these cycles and find the arc  $a^*$  with the bottleneck tolerance  $u^*$  and its cycle  $r^*$  (function COMPUTE-TOLERANCES( $c, f, R, a^*, r^*, u$ ), Algorithm 4). Note that among the cycles having the same maximal tolerance  $u^*$  we choose the one having the minimal size (number of arcs)  $|r|$  because it provides less branches

---

**Algorithm 3** Checks if cyclic routes in  $R$  are feasible, removes feasible cycles
 

---

```

function IS-FEASIBLE( $R$ )
  for  $r \in R$  do                                ▷ for each of the cyclic routes (cycles)  $R$ 
    if  $0 \notin r$  then
      continue                                ▷ no depot in cycle  $r$ 
    end if
     $d_r = \sum_{i \in r} d_i$                         ▷ total demand of clients in cycle  $r$ 
    if  $d_r > C$  or  $d_r < d_{\min}$  then
      continue                                ▷ overloaded or underloaded cycle
    end if
     $R = R \setminus \{r\}$                         ▷ remove feasible cycle  $r$ 
  end for
  if  $R = \emptyset$  then
    return true
  else
    return false
  end if
end function

```

---

in the search tree. Since calculation of tolerances (function COMPUTE-UPPER-TOLERANCE( $c, f, a$ ), Algorithm 5) is computationally difficult we stop the calculation of tolerances in non-bottleneck cycles as soon as possible. A cycle cannot have a bottleneck arc if it has an arc which tolerance is less than the currently maximal cycle tolerance  $u^*$  (or it is equal to  $u^*$  and the size of the cycle  $|r|$  is not less than the size of the cycle  $|r^*|$  having the currently maximal tolerance).

After the bottleneck cycle is found, we consequently remove one of its arcs in a while-loop and recursively call BRANCH-AND-BOUND( $c$ ) function for this branch. To remove an arc from the AP solution, we set its cost to  $+\infty$  and to forcibly include an arc to the AP solution we set its cost to  $-\infty$ . Let the bottleneck cycle arcs  $a_1, a_2, \dots, a_m$  be ordered by its tolerances:  $u_{a_1} \leq u_{a_2} \leq \dots \leq u_{a_m}$ . Then we have the following  $m$  branches:  $\overline{a_1}, a_1 \overline{a_2}, a_1 a_2 \overline{a_3}, \dots, a_1 a_2 \dots a_{m-1} \overline{a_m}$ . After all the branches are considered, we restore the original values of arc costs and return from recursion.

## 4 Computational Results

We perform our computational experiments on Intel Core i7 laptop machine with 1.9 GHz CPU and 8 Gb of memory. The search tree size (number of nodes) and computational time in seconds are shown in Table 1 for the classical cost-based branching rule and for the suggested tolerance-based one. In average, the search tree size is 45 times smaller for the suggested branching strategy. The computational time is 2.8 times smaller in average. Though for some instances, the running time

---

**Algorithm 4** Finds upper tolerances  $u$ , bottleneck arc  $a^*$  and cycle  $r^*$ 


---

```

function COMPUTE-TOLERANCES( $c, f, R, a^*, r^*, u$ )
   $u^* = -1$  ▷ bottleneck tolerance
   $u_{\min} = +\infty$  ▷ minimal tolerance in a cycle
   $a_{\min} = -1$  ▷ the arc with the minimal tolerance in a cycle
  for  $r \in R$  do ▷ for each cycle  $r$  of the infeasible cycles  $R$ 
    for  $a \in r$  do ▷ for each arc  $a$  in cycle  $r$ 
       $u_a = \text{COMPUTE-UPPER-TOLERANCE}(c, f, a)$ 
      if  $(u_a < u^*)$  or  $(u_a = u^* \text{ and } |r| \geq |r^*|)$  then ▷ skip non-bottleneck cycle
         $u_{\min} = +\infty$ 
        break
      end if
      if  $u_a < u_{\min}$  then
         $u_{\min} = u_a$ 
         $a_{\min} = a$ 
      end if
    end for
    if  $u_{\min} = +\infty$  then
      continue ▷ skip non-bottleneck cycle
    end if
    if  $(u_{\min} > u^*)$  or  $(u_{\min} = u^* \text{ and } |r| < |r^*|)$  then
       $u^* = u_{\min}$ 
       $r^* = r$ 
       $a^* = a_{\min}$ 
    end if
  end for
end function

```

---



---

**Algorithm 5** Computes the upper tolerance of arc  $a$ 


---

```

function COMPUTE-UPPER-TOLERANCE( $c, f, a$ )
   $c_a^0 = c_a$ 
   $c_a = +\infty$ 
   $f_{-a} = \text{JV}(c, x)$ 
   $c_a = c_a^0$ 
   $u_a = f_{-a} - f$ 
  return  $u_a$ 
end function

```

---

is slightly greater, for others it is considerably smaller. For example, it is 3 times smaller for FTV64 and 15 times smaller for FTV70 instances. The reduction in the search tree size varies from 2.8 times to 350 times.

**Table 1** Search tree size and running time for cost-based and tolerance-based branching

$ V $	Optimal value	Cost-based, nodes	Tolerance-based, nodes	Cost-based, sec	Tolerance-based, sec
33	1406	71555	5640	0.8	0.9
35	1644	137171	47492	2.2	5.4
38	1654	225737	29081	4.6	7.2
44	1740	1832873	88619	36.7	27.7
47	1891	151697	25863	3.5	7.8
55	1739	1324700	122381	41.2	52.1
64	1974	13981245	175153	523.2	173.5
70	2054	5534172	15797	194.1	12.2

## 5 Conclusion

The suggested tolerance-based branching strategy allows us to reduce the search tree size in a branch-and-bound algorithm considerably. In this chapter, we have presented preliminary results for a simple branch-and-bound algorithm without any efficient lower bounds and heuristic to find high-quality feasible ACVRP solutions. For example, one of the best exact algorithms for the ACVRP—the FTV algorithm by Fischetti et al. (1994) [5] computes the so-called disjunctive lower bound, then the lower bound based on min-cost flow relaxation, and finally applies the additive approach to get a tighter lower bound from these two bounds. The FTV algorithm also runs the VHVRP heuristic by Vigo (1996) [17] to obtain a feasible ACVRP solution. Our results for a full-fledged branch-and-bound algorithm with tight lower bounds and high-quality heuristic solutions will be published in a journal paper.

Since the suggested tolerance-based branch-and-bound algorithm requires solving of a great number of assignment problems (AP), it is reasonable to reduce the computational time spent for solving the AP. We apply one of the most efficient algorithms for the AP—the JV algorithm by Jonker and Volgenant (1987) [11]. Another idea is to use the algorithm of Volgenant (2006) [18] to compute an optimal AP solution and all its tolerances in  $O(n^3)$  time. We can also try to reduce the quantity of solved assignment problems by selecting the smallest cycle and the arc with the minimal tolerance in it instead of looking for the bottleneck cycle (see Turkensteen et al. (2008) [16]). Another source of research is to check whether the lower tolerances will be helpful either to connect unserved cycles with the so called underloaded feasible cycles (see Germs et al. (2012) [8]).

**Acknowledgements** The authors are supported by LATNA Laboratory, National Research University Higher School of Economics (NRU HSE), Russian Federation government grant, ag. 11.G34.31.0057.

Mikhail Batsyn and Boris Goldengorin were supported by the project No11-04-0008: “Calculus of tolerances in combinatorial optimization: theory and algorithms”, carried out within the Higher School of Economics 2011–2012 Academic Fund Program.

Mikhail Batsyn and Anton Kocheturov are supported by Federal Grant-in-Aid Program “Research and development on priority directions of development of the scientific-technological complex of Russia for 2007–2013” (Governmental Contract No. 14.514.11.4065).

## References

1. Baldacci, R., Hadjiconstantinou, E., Mingozzi, A.: An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Oper. Res.* **52**, 723–738 (2004)
2. Baldacci, R., Christofides, N., Mingozzi, A.: An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Math. Program., Ser. A* **115**(2), 351–385 (2008)
3. Baldacci, R., Mingozzi, A., Roberti, R.: New route relaxation and pricing strategies for the vehicle routing problem. *Oper. Res.* **59**(5), 1269–1283 (2011)
4. Baldacci, R., Mingozzi, A., Roberti, R.: Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *Eur. J. Oper. Res.* **218**, 1–6 (2012)
5. Fischetti, M., Toth, P., Vigo, D.: A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Oper. Res.* **42**, 846–859 (1994)
6. Fukasawa, R., Longo, H., Lysgaard, J., Poggi de Aragao, M., Reis, M., Uchoa, E., Werneck, R.: Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Math. Program., Ser. A* **106**, 491–511 (2006)
7. Gal, T.T., Greenberg, H.J. (eds.): *Advances in Sensitivity Analysis and Parametric Programming*. Internat. Ser. Oper. Res. Management Sci., vol. 6. Kluwer Academic, Boston (1997)
8. Germs, R., Goldengorin, B., Turkensteen, M.: Lower tolerance-based Branch and Bound algorithms for the ATSP. *Comput. Oper. Res.* **39**(2), 291–298 (2012)
9. Golden, B.L., Raghavan, S., Wasil, E.A.: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Operations Research/Computer Science Interfaces, vol. 43. Springer, New York (2008)
10. Goldengorin, B., Sierksma, G., Turkensteen, M.: Tolerance based algorithms for the ATSP. In: *Lecture Notes in Computer Science*, vol. 3353, pp. 222–234 (2004)
11. Jonker, R., Volgenant, A.: A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* **38**, 325–340 (1987)
12. Laporte, G.: Fifty years of vehicle routing. *Transp. Sci.* **43**(4), 408–416 (2009)
13. Lysgaard, J., Letchford, A.N., Eglese, R.W.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Math. Program., Ser. A* **100**, 423–445 (2004)
14. Pessoa, A., de Aragao, M.P., Uchoa, E.: Robust branch-cut-and-price algorithms for vehicle routing problems. In: Golden, B., Raghavan, S., Wasil, E. (eds.) *The Vehicle Routing Problem: Latest Advances and New Challenges*. Operations Research/Computer Science Interfaces, vol. 43, pp. 297–325 (2008)
15. Toth, P., Vigo, D.: *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia (2002)
16. Turkensteen, M., Ghosh, D., Goldengorin, B., Sierksma, G.: Tolerance-based branch and bound algorithms for the ATSP. *Eur. J. Oper. Res.* **189**(3), 775–788 (2008)
17. Vigo, D.: A heuristic algorithm for the asymmetric capacitated vehicle routing problem. *Eur. J. Oper. Res.* **89**(1), 108–126 (1996)
18. Volgenant, A.: An addendum on sensitivity analysis of the optimal assignment. *Eur. J. Oper. Res.* **169**(1), 338–339 (2006)

# Lower and Upper Bounds for the Preemptive Single Machine Scheduling Problem with Equal Processing Times

Mikhail Batsyn, Boris Goldengorin, Pavel Sukhov, and Panos M. Pardalos

**Abstract** The preemptive single machine scheduling problem of minimizing the total weighted completion time with equal processing times and arbitrary release dates is one of the four single machine scheduling problems with an open computational complexity status. In this chapter we present lower and upper bounds for the exact solution of this problem based on the assignment problem. We also investigate properties of these bounds and worst-case behavior.

**Keywords** Single machine scheduling · Lower bound · Upper bound · Assignment problem · Weighted completion time · Equal processing times · Release dates

## 1 Introduction

The complexity status of many scheduling problems is known, but there are four single machine scheduling problems which computational complexity is an open question. In this chapter, we consider one of these four problems—the preemptive single machine scheduling problem of minimizing the total weighted completion time with equal processing times and arbitrary release dates. In the notation

---

M. Batsyn (✉) · P. Sukhov · P.M. Pardalos

Laboratory of Algorithms and Technologies for Network Analysis, National Research University Higher School of Economics, 136 Rodionova, Nizhny Novgorod, Russian Federation  
e-mail: [mbatsyn@hse.ru](mailto:mbatsyn@hse.ru)

P. Sukhov

e-mail: [pavelandreevith@gmail.com](mailto:pavelandreevith@gmail.com)

P.M. Pardalos

e-mail: [pardalos@ufl.edu](mailto:pardalos@ufl.edu)

B. Goldengorin · P.M. Pardalos

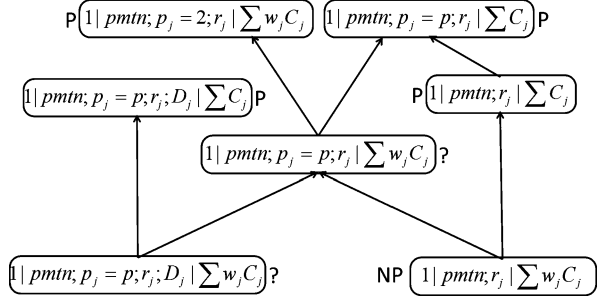
Center of Applied Optimization, University of Florida, 401 Weil Hall, P.O. Box 116595, Gainesville, FL 32611-6595, USA

B. Goldengorin

e-mail: [goldengorin@ufl.edu](mailto:goldengorin@ufl.edu)



**Fig. 1** Scheduling problems with completion time as an objective function and allowed preemptions



$\langle \text{machine environment} \mid \text{job characteristics} \mid \text{objective function} \rangle$  introduced by Graham et al. (1979) [6], this problem is written as follows.

$$1|pmtn; p_j = p; r_j | \sum w_j C_j \quad (1)$$

Here “1” means a single machine, “pmtn”—permutations are allowed: processing of a job can be interrupted in favor of another job,  $p_j = p$ —jobs have equal processing times  $p$ ,  $r_j$ —jobs have arbitrary release dates  $r_j$ ,  $\sum w_j C_j$ —the objective is to minimize the total weighted completion time, where  $w_j$  is the weight of job  $j$  and  $C_j$ —its completion time. There are no polynomial algorithms developed for this problem. So solving it for a large number of jobs or long processing time might be quite difficult.

Special cases of this problem are polynomially solvable, generalizations are NP-hard or have unknown complexity. Figure 1 shows the graph of existing single machine scheduling problems with completion time as an objective function and allowed preemptions. A good classification of a wide range of static scheduling problems is given by Herrman et al. (1993) [7].

The first of these problems  $1|pmtn; r_j | \sum C_j$  was solved by Baker (1974) [1]. He proved that this problem is polynomially solvable and, therefore, a special case of  $1|pmtn; p_j = p; r_j | \sum w_j C_j$  problem without weights:  $1|pmtn; p_j = p; r_j | \sum C_j$  is also polynomial. The generalization of this problem,  $1|pmtn; r_j | \sum w_j C_j$ , is strongly NP-hard. It was proved by Labetoulle et al. (1984) [8]. Another related problem having strict deadlines  $1|pmtn; p_j = p; r_j; D_j | \sum C_j$  is polynomial with complexity  $O(n \log(n))$  (Smith, 1956 [11]). Complexity of a more general  $1|pmtn; p_j = p; r_j; D_j | \sum w_j C_j$  problem is still an open question. Bouma and Goldengorin (2009) [3] proved that a special case of  $1|pmtn; p_j = p; r_j | \sum w_j C_j$  problem with  $p = 2$  is polynomially solvable. Obviously there is one more special case of  $1|pmtn; p_j = p; r_j | \sum w_j C_j$  problem: the problem with release dates equal to 0. But preemption has no sense without different release dates. In this case, the problem can be solved exactly using WSPT (Weighted Shortest Processing Time) rule (Pinedo, 2012 [10]).  $1 | \sum w_j C_j$  problem was one of the first problems in scheduling theory which complexity status was determined. Smith (1956) [11] proved that the complexity of this problem is  $O(n \log(n))$ . After that, in 1975  $1|r_j | \sum C_j$  problem was shown to be strongly NP-hard (Lenstra et al., 1975 [9]).

and therefore its weighted generalization  $1|r_j|\sum w_j C_j$  is NP-hard too. The special case of this generalization was solved by Baptiste (2000) [2]. He suggested a polynomial algorithm for  $1|p_j = p; r_j|\sum w_j C_j$  with complexity  $O(n^7)$ . Another generalization of  $1|p_j = p; r_j|\sum w_j C_j$  problem for parallel machines was shown to be polynomial by Brucker and Kravchenko (2008) [4].

The chapter is organized as follows. In the next section, we formulate the considered single machine problem and give an illustrating example. In the third and fourth sections, we describe the suggested lower and upper bounds based on the assignment problem. We provide the worst-case analysis of these bounds and give estimations of their precision. In the fifth section, we conclude the chapter with a short summary and some remarks on the future research.

## 2 Problem Formulation

The problem  $1|pmtn; p_j = p; r_j|\sum w_j C_j$  can be described as follows. We are given  $n \geq 1$  jobs that need to be processed on one machine. Jobs have the same processing time  $p_j = p$ , arbitrary release dates  $r_j$ , and weights  $w_j$ . A release date  $r_j$  is the time moment at which job  $j$  becomes available for processing. A weight  $w_j$  can be seen as a priority factor of job  $j$ . Release dates  $r_j$  are assumed to be non-negative integers, weights  $w_j$  and processing time  $p$  are assumed to be strictly positive integers. Preemptions are allowed, which means that processing of a job may be interrupted in favor of another job. The objective is to schedule the jobs such that the total weighted completion time  $\sum w_j C_j$  is minimized, where  $C_j$  denotes the completion time of job  $j$ . Also we assume that there are no idle time intervals. In what follows, we will call this scheduling problem shortly as *SP*.

To make the description clearer, let us consider the following example. Let  $n = 3$ ,  $p = 4$ ,  $w_j = 1, 2, 3$ ,  $r_j = 1, 4, 7$ . The feasible solutions of this problem are shown in Table 1 excluding the solutions in which a job with a greater weight is interrupted by a job with a smaller weight. Such solutions cannot be optimal.

The optimal solution is the first one in this example. Its total weighted completion time is  $w_1 \cdot C_1 + w_2 \cdot C_2 + w_3 \cdot C_3 = 1 \cdot 4 + 2 \cdot 8 + 3 \cdot 12 = 56$ . This solution can be represented as a vector  $(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3)$  which has  $np = 12$  elements corresponding to moments of time  $1, 2, \dots, 12$ . An element on position  $t$  shows what job (one part of a job) is scheduled to the time moment  $t$ .

## 3 Lower Bound

If we consider the solution of the scheduling problem *SP*, it can be noticed that it is actually an assignment of  $np$  job parts (each of  $n$  jobs has  $p$  parts) to  $np$  time moments  $1, 2, \dots, np$  such that every part of job  $j$  is assigned to a time moment not earlier than its release date  $r_j$ . Only the last ( $p$ -th) part of every job  $j$  is taken into account in the objective function. The cost of its assignment to time  $t$  is equal

**Table 1** Possible solutions

1	2	3	4	5	6	7	8	9	10	11	12	
1				2				3				
$1 \cdot 4 + 2 \cdot 8 + 3 \cdot 12 = 56$												
1				2				3				2
$1 \cdot 4 + 2 \cdot 12 + 3 \cdot 11 = 61$												
1				2		3				2		
$1 \cdot 4 + 2 \cdot 12 + 3 \cdot 10 = 58$												
1		2				3				1		
$1 \cdot 12 + 2 \cdot 7 + 3 \cdot 11 = 59$												
1		2				3				2	1	
$1 \cdot 12 + 2 \cdot 11 + 3 \cdot 10 = 64$												

**Table 2** Assignment problem

	1	2	3	4	5	6	7	8	9	10	11	12
1 (job 1, part 1)	0	0	0	0	0	0	0	0	0	$\infty$	$\infty$	$\infty$
2 (job 1, part 2)	$\infty$	0	0	0	0	0	0	0	0	0	$\infty$	$\infty$
3 (job 1, part 3)	$\infty$	$\infty$	0	0	0	0	0	0	0	0	0	$\infty$
4 (job 1, part 4)	$\infty$	$\infty$	$\infty$	4	5	6	7	8	9	10	11	12
5 (job 2, part 1)	$\infty$	$\infty$	$\infty$	0	0	0	0	0	0	$\infty$	$\infty$	$\infty$
6 (job 2, part 2)	$\infty$	$\infty$	$\infty$	$\infty$	0	0	0	0	0	0	$\infty$	$\infty$
7 (job 2, part 3)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	0	0	0	0	0	0	$\infty$
8 (job 2, part 4)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	14	16	18	20	22	24
9 (job 3, part 1)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	0	0	0	$\infty$	$\infty$	$\infty$
10 (job 3, part 2)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	0	0	0	$\infty$	$\infty$
11 (job 3, part 3)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	0	0	0	$\infty$
12 (job 3, part 4)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	30	33	36

to  $w_{jt}$ , and the costs for parts  $1, 2, \dots, p-1$  are all equal to 0. The objective is to minimize the total cost over all jobs. Thus,  $SP$  problem is equivalent to the linear assignment problem with additional constraints related to release dates  $r_j$  and also to the sequence of job parts which requires the last part of a job to be assigned to the latest time moment among the time moments of all its parts. Moreover, release date constraints can be taken into account in the classical assignment problem using infinite costs for the time moments to which job parts cannot be assigned. For the example in Table 1, the assignment problem will have the cost matrix shown in Table 2.

The solution is shown by gray cells. It satisfies all the constraints of the original  $SP$  problem except the sequence constraints requiring the last part of a job to be scheduled before all other parts. The following theorem shows that the assignment problem provides a lower bound on the optimal solution of  $SP$ . We will denote this assignment problem as  $AP_L$ .

**Theorem 1** *The solution of the following assignment problem  $AP_L$  gives a lower bound to the solution of the scheduling problem  $1|pmtn; p_j = p; r_j|\sum w_j C_j$ :*

$$\min \sum_{i=1}^{np} \sum_{t=1}^{np} c_{it} x_{it} \quad (2)$$

$$\sum_{t=1}^{np} x_{it} = 1 \quad \forall i = \overline{1, np} \quad (3)$$

$$\sum_{i=1}^{np} x_{it} = 1 \quad \forall t = \overline{1, np} \quad (4)$$

$$x_{it} \in \{0, 1\} \quad (5)$$

where

$$c_{it} = \begin{cases} 0, & \text{if } t \in [r_j + i', (n-1)p + 1 + i'] \text{ \& } i \bmod p > 0 \\ \infty, & \text{if } t \notin [r_j + i', (n-1)p + 1 + i'] \\ w_{i/p}t, & \text{if } t \in [r_j + p - 1, np] \text{ \& } i \bmod p = 0 \end{cases} \quad (6)$$

$$j = \lceil (i-1)/p \rceil + 1, \quad i' = (i-1) \bmod p$$

*Proof* We will show that  $SP$  problem is equivalent to the stated assignment problem  $AP_L$  with some additional constraints. By definition,  $SP$  is the problem of assigning every part  $1, 2, \dots, p$  of every job  $j = 1, 2, \dots, n$  to position (time moment)  $t = 1, 2, \dots, np$  of a schedule so that the total weighted completion time  $\sum_{j=1}^n w_j C_j$  is minimized and the following constraints are satisfied:

1. *Assignment constraints*: every part of every job should be assigned to exactly one position of a schedule and every position should be occupied by exactly one job part.
2. *Sequence constraints*: for every job  $j$  positions of its parts  $1, 2, \dots, p$  in a schedule should stay in the following sequence:  $t_1^j, t_2^j, \dots, t_p^j$ . Note that  $t_p^j = C_j$ .
3. *Release date constraints*: the first part of every job  $j$  should be assigned to the time moment which is not earlier than the release date of this job:  $t_1^j \geq r_j$ .

In the assignment problem  $AP_L$ , we minimize the following objective function:

$$f_{AP_L} = \sum_{i=1}^{np} \sum_{t=1}^{np} c_{it} x_{it} = \sum_{j=1}^n \sum_{t=1}^{np} c_{pj,t} x_{pj,t}$$

**Table 3**  $SP$  and  $AP_L$  solutions

1	p	2p	3p	np-p	np
2	n	n-1	...	3	1

1	p	p+1p+2				np
	2	n	n-1	...	3	...

because when  $i \bmod p > 0$  either  $c_{it} = 0$  or  $c_{it} = \infty$  and  $x_{it} = 0$ . Since  $x_{pj,t} = 1$  only for  $t = t_p^j$ ,  $c_{pj,t} = w_j t$ , and  $t_p^j = C_j$  then:

$$f_{AP_L} = \sum_{j=1}^n c_{pj,t_p^j} = \sum_{j=1}^n w_j t_p^j = \sum_{j=1}^n w_j C_j$$

So the objective function of  $AP_L$  is the same as for  $SP$ . *Assignment constraints* of  $SP$  are equivalent to constraints (3) and (4) of  $AP_L$ . *Release date constraints* of  $SP$  are always satisfied in  $AP_L$  solution because  $c_{p(j-1),t} = \infty$  for  $t < r_j$ , and thus  $t_1^j < r_j$  always gives  $f_{AP} = \infty$ . *Sequence constraints* of  $SP$  are not satisfied by  $AP_L$ . But these constraints can be represented by the following linear inequalities:

$$\forall j = \overline{1, n} \quad \forall t_0 = \overline{r_j + p - 1, np - 1} \quad \sum_{t=t_0+1}^{np} \sum_{i=1+p(j-1)}^{pj} x_{it} \leq p(1 - x_{pj,t_0}) \quad (7)$$

So we have got that  $AP_L$  is a relaxation of  $SP$  in which *sequence constraints* are removed. This proves that the optimal solution of  $AP_L$  gives a lower bound to the optimal solution of  $SP$ :  $f_{AP} \leq f_{SP}$ .  $\square$

Our goal is to determine how good is the lower bound given by the  $AP_L$ . For this purpose, we are going to find the worst case for this bound. The worst case is the scheduling problem for which the  $AP_L$  solution differs from the  $SP$  solution as much as possible.

**Theorem 2** *The worst case for the  $AP_L$  (for which the ratio  $f_{AP_L}/f_{SP}$  is minimal) is the scheduling problem with  $p < n$ ,  $w_1 \leq w_2 \leq \dots \leq w_n$ ,  $r_1 = r_2 = 1$ ,  $\forall j = \overline{3, n}$   $r_j = n - j + 2$ .*

*Proof* Here we provide only some ideas and a scheme of the proof. The complete and detailed proof will be presented in a full journal paper. For the described scheduling problem,  $SP$  and  $AP_L$  solutions are shown in Table 3.

The proof is based on the following observation. If in the optimal  $SP$  schedule job  $n$  with the maximal weight  $w_n$  starts from time  $(t + 1)$ , then it ends at time  $(t + p)$ , because the job with the maximal weight cannot be preempted in the optimal schedule. It is not difficult to prove that preemption of any job by a job with a smaller weight always increases the value of the objective function. It is also easy to prove

that in this case its release date  $r_n \geq t - p + 2$  if  $t \geq p$  (if  $t < p$ , then  $r_n$  can have any value and job  $n$  in the  $AP_L$  solution will be close to its position in the  $SP$  solution). So in the optimal  $AP_L$  schedule the last part of job  $n$  will be placed at the earliest possible time  $r_n + p - 1 \geq t + 1$ . The relative difference of the objective functions of  $SP$  and  $AP_L$  solutions only for job  $n$  (assuming that all other jobs last parts are placed on the same positions in  $AP_L$  and  $SP$  solutions) is as follows.

$$\frac{f_{SP} - f_{AP_L}}{f_{SP}} = \frac{((t + p) - (t + 1))w_n}{(t + p)w_n} = \frac{p - 1}{t + p}$$

This difference is maximal when  $t$  is minimal. Since  $t \geq p$  then for job  $n$  the worst case of  $AP_L$  solution is realized when  $t = p$ . In this case release date  $r_n = 2$ , job  $n$  starts at time  $p + 1$  and ends at time  $2p$  in the optimal  $SP$  schedule. Since job  $n$  makes the greatest contribution to the objective function it is reasonable to consider it first, then job  $n - 1$ , and so on. Following this logic after freezing the position of job  $n$  in the optimal  $SP$  schedule, we find that in the worst case job  $n - 1$  starts at time  $2p + 1$ , ends at time  $3p$ , and  $r_{n-1} = 3$ . An so on till job 3 which should start at time  $np - 2p + 1$ , end at time  $np - p$ , and have release date  $r_3 = n - 1$  (see Table 3). The remaining jobs 2 and 1 can have release dates  $r_1 = r_2 = 1$  and can be placed to time intervals  $[1, p]$  and  $[np - p + 1, np]$  in the optimal  $SP$  schedule.  $\square$

The worst case for the  $AP_L$  solution immediately gives us the estimation of the  $AP_L$  lower bound precision.

**Theorem 3** *The  $AP_L$  optimal solution provides a value of the objective function  $f_{AP_L}$  which is not less than  $(\frac{1}{p} + \frac{2}{n+1} - \frac{2}{p(n+1)})f_{SP}$ .*

*Proof* According to Theorem 2, the worst case for  $AP_L$  is the scheduling problem with  $p < n$ ,  $w_1 \leq w_2 \leq \dots \leq w_n$ ,  $r_1 = r_2 = 1$ ,  $\forall j = 3, n$   $r_j = n - j + 2$ . The  $AP_L$  and  $SP$  solutions for this problem are shown in Table 3. The values of the objective function and the relative difference are as follows.

$$\begin{aligned} f_{SP} &= \sum_{i=1}^{n-2} (i+1)pw_{n+1-i} + pw_2 + npw_1 \\ f_{AP_L} &= \sum_{i=1}^{n-2} (p+i)w_{n+1-i} + pw_2 + npw_1 \\ \frac{f_{SP} - f_{AP_L}}{f_{SP}} &= \frac{\sum_{i=1}^{n-2} i(p-1)w_{n+1-i}}{\sum_{i=1}^{n-2} (i+1)pw_{n+1-i} + pw_2 + npw_1} \end{aligned}$$

The difference is maximal when  $w_1 = w_2 = 0$ , and so we have:

$$\frac{f_{SP} - f_{AP_L}}{f_{SP}} \leq \frac{\sum_{i=1}^{n-2} i(p-1)w_{n+1-i}}{\sum_{i=1}^{n-2} (i+1)pw_{n+1-i}}$$

Let us consider this ratio only for one job  $i$  without others.

$$\frac{i(p-1)w_{n+1-i}}{(i+1)pw_{n+1-i}} = \frac{ip-i}{ip+p} = 1 - \frac{1}{p} - \frac{p-1}{(i+1)p}$$

This value is maximal when  $i$  is maximal:  $i = n-2$ . This means that the  $(n-2)$ -th summand makes the greatest contribution to the value of the whole expression. If all the weights except  $w_3$  are zero, then we will get the maximum, but  $w_3 \leq w_4 \leq \dots \leq w_n$ . So to make the contribution of the last summand as great as possible we should set all the weights except  $w_3$  as small as possible:  $w_n = w_{n-1} = \dots = w_3$ . In this case, we have:

$$\begin{aligned} \frac{f_{SP} - f_{AP_L}}{f_{SP}} &= \frac{(p-1)w_3 \sum_{i=1}^{n-2} i}{pw_3 \sum_{i=1}^{n-2} (i+1)} = \frac{(p-1)(n-1)}{p(n+1)} \\ &= 1 - \left( \frac{1}{p} + \frac{2}{n+1} - \frac{2}{p(n+1)} \right) \end{aligned}$$

These are only intuitive considerations which lead to the estimation for the lower bound. But this estimation should be carefully proved. First, we need to prove the following auxiliary inequality.

$$\sum_{i=1}^{n-2} (i+1)pw_{n+1-i} \geq \frac{1}{2}p(n-2)(n+1)w_3 \quad (8)$$

Since  $\forall i = \overline{1, n-2} \ w_{n+1-i} \geq w_3$ , we have:

$$\sum_{i=1}^{n-2} (i+1)pw_{n+1-i} \geq \sum_{i=1}^{n-2} (i+1)pw_3 = \frac{1}{2}p(n-2)(n+1)w_3$$

Let us use the mathematical induction to prove the estimation for the lower bound:

$$\frac{f_{SP} - f_{AP_L}}{f_{SP}} \leq \frac{\sum_{i=1}^{n-2} i(p-1)w_{n+1-i}}{\sum_{i=1}^{n-2} (i+1)pw_{n+1-i}} \leq \frac{(p-1)(n-1)}{p(n+1)}$$

For  $n = 3$ , this inequality is true:

$$\frac{(p-1)w_3}{2pw_3} \leq \frac{2(p-1)}{4p}$$

Now we assume that the inequality is true for  $n = k$  and prove that then it is also true for  $n = k+1$ .

$$\frac{\sum_{i=1}^{k-1} i(p-1)w_{k+2-i}}{\sum_{i=1}^{k-1} (i+1)pw_{k+2-i}} = \frac{\sum_{i=1}^{k-2} i(p-1)w_{k+2-i} + (k-1)(p-1)w_3}{\sum_{i=1}^{k-2} (i+1)pw_{k+2-i} + kpw_3}$$

Let us make the following denotations.

$$a = \sum_{i=1}^{k-2} i(p-1)w_{k+2-i}$$

$$b = \sum_{i=1}^{k-2} (i+1)pw_{k+2-i}$$

Since our statement is true for  $n = k$ , we have:

$$\frac{a}{b} \leq \frac{(p-1)(k-1)}{p(k+1)} \implies a \leq \frac{(p-1)(k-1)}{p(k+1)}b$$

We need to prove that:

$$\frac{a + (k-1)(p-1)w_3}{b + kpw_3} \leq \frac{(p-1)k}{p(k+2)}$$

Using the statement for  $n = k$ , we have

$$\begin{aligned} \frac{a + (k-1)(p-1)w_3}{b + kpw_3} &\leq \frac{\frac{(p-1)(k-1)}{p(k+1)}b + (k-1)(p-1)w_3}{b + kpw_3} \\ &= \frac{p-1}{p} \cdot \frac{(k-1)b + (k-1)(k+1)pw_3}{(k+1)b + k(k+1)pw_3} \end{aligned}$$

So it is enough to prove that:

$$\begin{aligned} \frac{(k-1)b + (k-1)(k+1)pw_3}{(k+1)b + k(k+1)pw_3} &\leq \frac{k}{(k+2)} \\ \iff (k+2)(k-1)b + (k+2)(k-1)(k+1)pw_3 &\leq k(k+1)b + k^2(k+1)pw_3 \\ \iff -2b &\leq (k+1)(k-2)pw_3 \\ \iff b &\geq \frac{1}{2}p(k-2)(k+1)w_3 \end{aligned}$$

Since the auxiliary inequality (8) is true for any  $n$  and any weights  $w_j$ , we have the inequality which we need to prove.

$$b = \sum_{i=1}^{k-2} (i+1)pw_{k+2-i} \geq \frac{1}{2}p(k-2)(k+1)w_3$$

This proves the inductive step and so our statement is true.

$$\frac{f_{SP} - f_{APL}}{f_{SP}} \leq \frac{(p-1)(n-1)}{p(n+1)} = 1 - \left( \frac{1}{p} + \frac{2}{n+1} - \frac{2}{p(n+1)} \right)$$



**Table 4**  $SP$  and  $AP_L$  solutions for a worst case example

1	4	8	12	392	396
2	99	98	...	3	1

1	4	5	6			396
	2	99	98	...	3	...

$$\Rightarrow \frac{f_{AP_L}}{f_{SP}} \geq \frac{1}{p} + \frac{2}{n+1} - \frac{2}{p(n+1)}$$

This ends the proof of the estimation for the lower bound  $f_{AP_L}$ .  $\square$

Let us show the worst case for the  $AP_L$  solution on an example with  $n = 99$  jobs,  $p = 4$ ,  $r_1 = r_2 = 1$ ,  $r_{99} = 2$ ,  $r_{98} = 3, \dots, r_3 = 98$ , and weights  $w_1 = \epsilon$ ,  $w_j = 1 + \epsilon j$  for  $j = 2, 99$ , where  $\epsilon$  is a very small value. The optimal  $SP$  and  $AP_L$  solutions for this problem are shown in Table 4.

Neglecting the value of  $\epsilon$ , we have  $f_{SP} = 4 + 8 + \dots + 4 \cdot 98 = 19404$ ,  $f_{AP_L} = 4 + 5 + \dots + 101 = 5145$ . The ratio of  $f_{AP_L}$  to  $f_{SP}$  and its estimation are the following.

$$\begin{aligned} \frac{f_{AP_L}}{f_{SP}} &= \frac{5145}{19404} \approx 0.26515 \\ \frac{f_{AP_L}}{f_{SP}} &\geq \frac{1}{p} + \frac{2}{n+1} - \frac{2}{p(n+1)} = \frac{1}{4} + \frac{2}{100} - \frac{2}{400} = 0.265 \end{aligned}$$

As one can see in the worst case (when  $n$  is big)  $AP_L$  lower bound can be  $p$  times smaller than the optimal  $SP$  solution  $f_{AP_L}$ .

## 4 Upper Bound

The main drawback of the assignment problem  $AP_L$  is that it does not include the sequence constraints. However, it is possible to incorporate the sequence constraints to the assignment problem. For this purpose, we set the assignment costs  $c_{it}$  to be equal to  $w_{[(i-1)/p]+1}t$  not only for the last part of a job (when  $i \bmod p = 0$ ) but for all its parts. For the example from Table 1, the assignment problem will have the cost matrix shown in Table 5.

The optimal assignment problem solution is highlighted with gray color. Now it satisfies all the constraints of the original  $SP$  problem. Unfortunately, it is not an optimal solution for the  $SP$  problem, because its objective function differs from the scheduling problem  $SP$  objective function. The following theorem proves that this assignment problem provides an upper bound for the optimal solution of  $SP$ . We will denote this assignment problem as  $AP_U$ .

**Table 5** Assignment problem

	1	2	3	4	5	6	7	8	9	10	11	12
1 (job 1, part 1)	1	2	3	4	5	6	7	8	9	$\infty$	$\infty$	$\infty$
2 (job 1, part 2)	$\infty$	2	3	4	5	6	7	8	9	10	$\infty$	$\infty$
3 (job 1, part 3)	$\infty$	$\infty$	3	4	5	6	7	8	9	10	11	$\infty$
4 (job 1, part 4)	$\infty$	$\infty$	$\infty$	4	5	6	7	8	9	10	11	12
5 (job 2, part 1)	$\infty$	$\infty$	$\infty$	8	10	12	14	16	18	$\infty$	$\infty$	$\infty$
6 (job 2, part 2)	$\infty$	$\infty$	$\infty$	$\infty$	10	12	14	16	18	20	$\infty$	$\infty$
7 (job 2, part 3)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	12	14	16	18	20	22	$\infty$
8 (job 2, part 4)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	14	16	18	20	22	24
9 (job 3, part 1)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	21	24	27	$\infty$	$\infty$	$\infty$
10 (job 3, part 2)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	24	27	30	$\infty$	$\infty$
11 (job 3, part 3)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	27	30	33	$\infty$
12 (job 3, part 4)	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	30	33	36

**Theorem 4** The solution of the following assignment problem  $AP_U$  gives an upper bound to the solution of the scheduling problem  $1|pmtn; p_j = p; r_j| \sum w_j C_j$ .

$$\min \sum_{i=1}^{np} \sum_{t=1}^{np} c_{it} x_{it} \quad (9)$$

$$\sum_{t=1}^{np} x_{it} = 1 \quad \forall i = \overline{1, np} \quad (10)$$

$$\sum_{i=1}^{np} x_{it} = 1 \quad \forall t = \overline{1, np} \quad (11)$$

$$x_{it} \in \{0, 1\} \quad (12)$$

where

$$c_{it} = \begin{cases} w_j t, & \text{if } t \in [r_j + i', (n-1)p + 1 + i'] \\ \infty, & \text{if } t \notin [r_j + i', (n-1)p + 1 + i'] \end{cases} \quad (13)$$

$$j = \lceil (i-1)/p \rceil + 1, \quad i' = (i-1) \bmod p$$

*Proof* We provide here only the basic ideas of the proof. The complete detailed proof will be presented in a full journal paper. The ideas are the following.

1. In the  $AP_U$  solution, a job with a smaller weight being processed will be interrupted by a job with a greater weight immediately when this job becomes available.
2. In the  $AP_U$  solution, a job with a smaller weight never interrupts a job with a greater weight.
3. If in the  $AP_U$  solution job  $i$  is interrupted by job  $j$ , then job  $i$  will be completed after job  $j$ .

**Table 6**  $SP$  and  $AP_U$  solutions

1	p	2p					np
1	2	...				n	

1	p-1	2p-2			np-n-p+1	np-n+1	np-n+2	np-n+3	np-1	np
1	2	...	n-1	n	n-1	n-2	...	2	1	

4. The  $AP_U$  solution can be obtained by assigning at every time moment the available job with the greatest weight.

It follows from statement 4 that the  $AP_U$  solution is a feasible solution for the  $SP$  problem. So the value of the  $SP$  objective function for this solution  $f_{AP_U}$  is an upper bound for the optimal value  $f_{SP}$  of this objective function.  $\square$

We want to determine how good is the upper bound given by the  $AP_U$ . For this purpose, we are going to find the worst case for this bound. The worst case is the scheduling problem for which the  $AP_U$  solution differs from the  $SP$  solution as much as possible in terms of the  $SP$  objective function.

**Theorem 5** *The worst case for the  $AP_U$  (for which the ratio  $f_{AP_U}/f_{SP}$  is maximal) is the scheduling problem with  $p > n$ ,  $w_1 \leq w_2 \leq \dots \leq w_n$ ,  $\forall j = \overline{1, n}$   $r_j = (p-1)(j-1) + 1$ .*

*Proof* Here we provide only some ideas and a scheme of the proof. The complete and detailed proof will be presented in a full journal paper. For the described scheduling problem,  $SP$  and  $AP_U$  solutions are shown in Table 6. The proof is based on the following statements.

1. If in the  $SP$  solution job  $i$  is interrupted by job  $j$ , then job  $i$  will be completed after job  $j$ .
2. If in the  $SP$  solution job  $i$  is interrupted by job  $j$ , then it is made in time moment  $r_j$ .
3. In the  $SP$  solution, the job which first part is assigned to some time moment has the greatest weight among all the jobs available at this moment and not started yet.
4. If in the  $SP$  solution job  $i$  is interrupted by job  $j$ , then the same is true for the  $AP_U$  solution.
5. If in the  $SP$  solution whole job  $j$  is scheduled before whole job  $i$  with a smaller weight, then the same is true for the  $AP_U$  solution.

From statements 4 and 5, it follows that the greatest difference between the  $AP_U$  and  $SP$  solutions is reached when the  $SP$  solution does not have interrupts.  $\square$

The worst case for the  $AP_U$  solution immediately gives us the estimation of the  $AP_U$  upper bound precision.

**Theorem 6** *The  $AP_U$  optimal solution provides a value of the SP objective function  $f_{AP_U}$  which is not greater than  $(2 - \frac{1}{p} - \frac{2}{n+1} + \frac{2}{p(n+1)})f_{SP}$ .*

*Proof* According to Theorem 5, the worst case for  $AP_U$  is the scheduling problem with  $p > n$ ,  $w_1 \leq w_2 \leq \dots \leq w_n$ ,  $\forall j = \overline{1, n}$   $r_j = (p-1)(j-1) + 1$ . The  $AP_U$  and  $SP$  solutions for this problem are shown in Table 6. The values of the objective function and the ratio are as follows.

$$\begin{aligned} f_{SP} &= \sum_{i=1}^n ipw_i \\ f_{AP_U} &= \sum_{i=1}^n (np+1-i)w_i \\ \frac{f_{AP_U}}{f_{SP}} &= \frac{\sum_{i=1}^n (np+1-i)w_i}{\sum_{i=1}^n ipw_i} \end{aligned}$$

Let us consider this ratio only for one job  $i$  without others.

$$\frac{(np+1-i)w_i}{ipw_i} = \frac{np+1-i}{ip} = -\frac{1}{p} + \frac{np+1}{ip}$$

This value is maximal when  $i$  is minimal:  $i = 1$ . This means that the first summand makes the greatest contribution to the value of the whole expression. If all the weights except  $w_1$  are zero, then we will get the maximum, but  $w_1 \leq w_2 \leq \dots \leq w_n$ . So to make the contribution of the first summand as great as possible we should set all the weights except  $w_1$  as small as possible:  $w_n = w_{n-1} = \dots = w_1$ . In this case, we have:

$$\frac{f_{AP_U}}{f_{SP}} = \frac{\sum_{i=1}^n (np+1-i)w_1}{\sum_{i=1}^n ipw_1} = \frac{2np-n+1}{p(n+1)} = 2 - \frac{1}{p} - \frac{2}{n+1} + \frac{2}{p(n+1)}$$

These are only intuitive considerations which lead to the estimation for the upper bound. But this estimation should be carefully proved. Let us use the mathematical induction to prove the following inequality.

$$\frac{f_{AP_U}}{f_{SP}} = \frac{\sum_{i=1}^n (np+1-i)w_i}{\sum_{i=1}^n ipw_i} \leq \frac{2np-n+1}{p(n+1)}$$

For  $n = 1$ , our statement is true:

$$\frac{f_{AP_U}}{f_{SP}} = \frac{(p+1-1)w_1}{pw_1} = 1, \quad \frac{2np-n+1}{p(n+1)} = \frac{2p}{2p} = 1$$

Assuming that the statement is true for  $n = k$ :

$$\frac{\sum_{i=1}^k (kp+1-i)w_i}{\sum_{i=1}^k ipw_i} \leq \frac{2kp-k+1}{p(k+1)}$$

we need to prove that it is also true for  $n = k + 1$ :

$$\frac{\sum_{i=1}^{k+1} ((k+1)p + 1 - i)w_i}{\sum_{i=1}^{k+1} ipw_i} \leq \frac{2(k+1)p - k}{p(k+2)}$$

We use the following denotations.

$$a = \sum_{i=1}^k (kp + 1 - i)w_i$$

$$b = \sum_{i=1}^k ipw_i$$

Since we assume that the statement is true for  $n = k$ , we have:

$$\frac{a}{b} \leq \frac{2kp - k + 1}{p(k+1)} \implies a \leq \frac{2kp - k + 1}{p(k+1)}b$$

For  $n = k + 1$ , we have:

$$\begin{aligned} & \frac{\sum_{i=1}^{k+1} ((k+1)p + 1 - i)w_i}{\sum_{i=1}^{k+1} ipw_i} \\ &= \frac{\sum_{i=1}^k (kp + 1 - i)w_i + \sum_{i=1}^k pw_i + ((k+1)p - k)w_{k+1}}{\sum_{i=1}^k ipw_i + (k+1)w_{k+1}} \\ &= \frac{a + \sum_{i=1}^k pw_i + ((k+1)p - k)w_{k+1}}{b + (k+1)w_{k+1}} \\ &\leq \frac{\frac{2kp - k + 1}{p(k+1)}b + \sum_{i=1}^k pw_i + ((k+1)p - k)w_{k+1}}{b + (k+1)w_{k+1}} \end{aligned}$$

So we need to prove the following:

$$\begin{aligned} & \frac{(2kp - k + 1)b + p(k+1) \sum_{i=1}^k pw_i + p(k+1)((k+1)p - k)w_{k+1}}{p(k+1)(b + (k+1)w_{k+1})} \\ &\leq \frac{2(k+1)p - k}{p(k+2)} \end{aligned}$$

Multiplying this inequality by the denominators, we get an equivalent one:

$$\begin{aligned} & (p(k+2)(2kp - k + 1) - 2(k+1)^2p^2 + k(k+1)p)b \\ & + p^2(k+1)(k+2) \sum_{i=1}^k pw_i + p^2(k+1)(k+2)((k+1)p - k)w_{k+1} \end{aligned}$$

$$\begin{aligned}
& -2p^3(k+1)^3w_{k+1} + p^2k(k+1)^2w_{k+1} \leq 0 \\
\iff & (2p - 2p^2)b + p^2(k+1) \left( (k+2) \sum_{i=1}^k pw_i - (k^2p + kp + k)w_{k+1} \right) \\
& \leq 0
\end{aligned}$$

Substituting the expression for  $b$  we come to the following inequality which we need to prove:

$$\begin{aligned}
& k(k+1)(kp + p + 1)w_{k+1} - \left( p(k+1)(k+2) \sum_{i=1}^k w_i - 2(p-1) \sum_{i=1}^k iw_i \right) \geq 0 \\
\iff & k(k+1)(kp + p + 1)w_{k+1} - \left( \sum_{i=1}^k (p(k+1)(k+2) - 2(p-1)i)w_i \right) \\
& \geq 0
\end{aligned}$$

It is clear that  $p(k+1)(k+2) - 2(p-1)i \geq 0$  and so the maximum value of the sum in the brackets is reached when  $w_i = w_{k+1}$  (because  $w_i \leq w_{k+1}$ ). Thus, we have:

$$\begin{aligned}
& k(k+1)(kp + p + 1)w_{k+1} - \left( \sum_{i=1}^k (p(k+1)(k+2) - 2(p-1)i)w_i \right) \\
& \geq k(k+1)(kp + p + 1)w_{k+1} - \left( \sum_{i=1}^k (p(k+1)(k+2) - 2(p-1)i)w_{k+1} \right) \\
& = k(k+1)(kp + p + 1)w_{k+1} - (p(k+1)(k+2)k - (p-1)k(k+1))w_{k+1} \\
& = k(k+1)(kp + p + 1 - kp - 2p + p - 1)w_{k+1} = 0
\end{aligned}$$

This proves the inductive step and so our statement is true.

$$\frac{f_{AP_U}}{f_{SP}} \leq \frac{2np - n + 1}{p(n+1)} = 2 - \frac{1}{p} - \frac{2}{n+1} + \frac{2}{p(n+1)}$$

This ends the proof of the estimation for the upper bound  $f_{AP_U}$ .  $\square$

We show the worst case for the  $AP_U$  solution on an example with  $n = 99$  jobs,  $p = 100$ ,  $r_1 = 1, r_2 = 100, r_3 = 199, \dots, r_{99} = 9703$ , and weights  $w_j = 1 + \epsilon j$  for  $j = \overline{1, 99}$ , where  $\epsilon$  is a very small value. The optimal  $SP$  and  $AP_L$  solutions for this problem are shown in Table 7.

Neglecting the value of  $\epsilon$  we have  $f_{SP} = 100 + 200 + \dots + 100 \cdot 99 = 495000$ ,  $f_{AP_U} = 9802 + 9803 + \dots + 9900 = 975249$ . The ratio of  $f_{AP_U}$  to  $f_{SP}$  and its esti-

**Table 7**  $SP$  and  $AP_L$  solutions for a worst case example

1	100	200	9900						
1	2	...	99						

1	99	198	9702	9802	9803	9804	9899	9900	
1	2	...	98	99	98	97	...	2	1

mation are the following.

$$\frac{f_{AP_U}}{f_{SP}} = \frac{975249}{495000} = 1.9702$$

$$\frac{f_{AP_U}}{f_{SP}} \leq 2 - \frac{1}{p} - \frac{2}{n+1} + \frac{2}{p(n+1)} = 2 - \frac{1}{100} - \frac{2}{100} + \frac{2}{10000} = 1.9702$$

As one can see in the worst case  $AP_U$  upper bound cannot be more than two times greater than the optimal value  $f_{SP}$ .

## 5 Concluding Remarks

In this chapter, we have studied the relationships between the linear assignment problem (AP) and the preemptive single machine scheduling problem  $1|pmtn; p_j = p; r_j| \sum w_j C_j$  of minimizing the total weighted completion time with equal processing times and arbitrary release dates (further abbreviated as  $SP$ ). The relationships between these problems might be expressed as follows.  $SP$  is equivalent to the modified AP with adjusted assignment costs (reflecting the release dates of the jobs) and additional sequence constraints. We have derived lower and upper bounds based on the relationships between the AP and  $SP$ . Our theoretical analysis of the lower bound shows that in the case with equal processing times  $p = 2$  and infinitely many jobs the lower bound is just a half ( $1/2$ ) of the unknown optimal value  $f_{SP}$  of the objective function for  $SP$ . And it tends to become worse when the processing time  $p$  is increased. Meanwhile for the same case with  $p = 2$  and infinitely many jobs the AP-based upper bound tends to come as close as possible to  $3/2 f_{SP}$ . And it tends to reach  $2 f_{SP}$  value when the processing time  $p$  is increased. Note that the best known approximation ratio for a similar NP-hard problem with arbitrary processing times evaluates by 1.47 (see Goemans et al. (2000) [5]).

The first direction of our future research is to empirically compare the quality of our upper bound and the 1.47 upper bound of Goemans et al. (2000) [5] algorithm for the so called equal binary processing times  $p = 2$  reflecting the most popular in practice case of scheduling jobs with two essential units: the start unit and the completion unit. Another direction of our research is to derive similar AP-based upper and lower bounds for other single machine scheduling problems including the NP-hard version of  $SP$  with arbitrary processing times (see Lenstra et al. (1997) [9]).

**Acknowledgements** The authors are partially supported by LATNA Laboratory, National Research University Higher School of Economics (NRU HSE), Russian Federation government grant, ag. 11.G34.31.0057.

Boris Goldengorin's research was partially supported by the Exchange Visiting Program Number P-1-01285 carried out at the Center of Applied Optimization, University of Florida, USA.

## References

1. Baker, K.R.: Introduction to Sequencing and Scheduling. Wiley, New York (1974), 305 pp.
2. Baptiste, P.: Scheduling equal-length jobs on identical parallel machines. *Discrete Appl. Math.* **103**(1), 21–32 (2000)
3. Bouma, H.W., Goldengorin, B.: A polytime algorithm based on a primal LP model for the scheduling problem  $1|pmtn; p_j = 2; r_j|\sum w_j C_j$ . Proceedings of the 2010 American Conference on Applied Mathematics, AMERICAN-MATH'10, Stevens Point, Wisconsin, USA, pp. 415–420. World Scientific and Engineering Academy and Society (WSEAS) (2010)
4. Brucker, P., Kravchenko, S.A.: Scheduling jobs with equal processing times and time windows on identical parallel machines. *J. Sched.* **11**, 229–237 (2008)
5. Goemans, M.X., Wein, J.M., Williamson, D.P.: A 1.47-approximation algorithm for a preemptive single-machine scheduling problem. *Oper. Res. Lett.* **26**, 149–154 (2000)
6. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.*, **5**, 287–326 (1979)
7. Herrman, J., Lee, C., Snowden, J.: A classification of static scheduling problems. In: Pardalos, P.M. (ed.) *Complexity in Numerical Optimization*, pp. 203–253. World Scientific, Singapore (1993)
8. Labetoulle, J., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Preemptive scheduling of uniform machines subject to release dates. In: *Progress in Combinatorial Optimization*, pp. 245–261. Academic Press, Toronto (1984)
9. Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P.: Complexity of machine scheduling problems. studies in integer programming. In: Hammer, P.L., Johnson, E.L., Korte, B.H., Nemhauser, G.L. (eds.) *Annals of Discrete Mathematics*, vol. 1, pp. 343–362. North-Holland, Amsterdam (1977)
10. Pinedo, M.L.: *Scheduling: Theory, Algorithms, and Systems*, 4th edn. Springer, Berlin (2012), 673 pp.
11. Smith, W.E.: Various optimizers for single-stage production. *Nav. Res. Logist. Q.* **3**, 59–66 (1956)



# Comparative Analysis of Two Similarity Measures for the Market Graph Construction

Grigory A. Bautin, Valery A. Kalyagin, and Alexander P. Koldanov

**Abstract** Market graph is built on the basis of some similarity measure for financial asset returns. The chapter considers two similarity measures: classic Pearson correlation and sign correlation. We study the associated market graphs and compare the conditional risk of the market graph construction for these two measures of similarity. Our main finding is that the conditional risk for the sign correlation is much better than for the Pearson correlation for larger values of threshold for several probabilistic models. In addition, we show that for some model the conditional risk for sign correlation dominates over the conditional risk for Pearson correlation for all values of threshold. These properties make sign correlation a more appropriate measure for the maximum clique analysis.

**Keywords** Network market analysis · Market graph · Similarity measures · Pearson correlation · Sign correlation · Conditional risk

## 1 Introduction

Modeling is an important tool for the analysis of the processes taking place in the financial markets. In many models, a crucial role is played by the features of simultaneous changes of financial assets prices. Adopting some measure of similarity, one comes to an idea of considering the market as a complex network in which financial assets are its vertices, and the edges reflect the degree of connectivity of their behavior. This approach was initially proposed and developed in [1–3]. In these works, the weighted graph representing the financial market is studied using different techniques, such as minimum spanning tree (MST) analysis [1], or using the planar

---

G.A. Bautin (✉) · V.A. Kalyagin · A.P. Koldanov

Laboratory of Algorithms and Technologies for Networks Analysis, National Research University Higher School of Economics, 136 Rodionova, Nizhny Novgorod, Russian Federation  
e-mail: [gbautin@hse.ru](mailto:gbautin@hse.ru)

V.A. Kalyagin  
e-mail: [vkalyagin@hse.ru](mailto:vkalyagin@hse.ru)

A.P. Koldanov  
e-mail: [akoldanov@hse.ru](mailto:akoldanov@hse.ru)

maximally filtered graph (PMFG) [2]. These procedures can be commonly considered as filtering of a complex weighted graph in order to fetch the most important information, and proceeding to considering of a simpler subgraph. This approach is being actively developed, and there are lots of recent proceeding, for example see [3].

Another approach proposed and developed in [4–6] is constructing of the market graph. The procedure of its construction consists in comparing the values of pairwise similarity measure to a threshold value, and leaving only the edges for which the similarity measure exceeds the threshold. Analysis of the resulting graph produces interesting results concerning structural properties of the market.

The most popular measure of similarity of the random variables behavior is the classic Pearson correlation coefficient. This measure is well studied, and is known to have both strengths and weaknesses. It is well known that for a multivariate normal distribution, covariance matrix is a sufficient statistic [7]. However, the main drawback of the classical correlation is its weak robustness to failure of the assumption of normality. It is known [8] that in non-normal case the correlation matrix does not reflect the dependence of random variables.

In paper [9], sign correlation of Fechner is used as an alternative measure of similarity. This measure is based on the probability of coincidence of the random variables signs. It is shown that such a measure is appropriate for the market network analysis, has a simple interpretation, can be generalized to any number of random variables, and has a connection to the Pearson correlation in the case of normal distribution. In [9], this similarity measure is applied to the analysis of Russian and Sweden financial markets.

The main goal of the present chapter is the comparative analysis of Pearson correlation and sign correlation in application to the market graph construction for different probabilistic models. We use the concepts of true and sample market graph defined in [10], and study the conditional risk of sample market graph construction for two measures of similarity—classic Pearson coefficient and sign correlation combined with several probabilistic models of the market. Our main result is finding that the conditional risk for the sign correlation is much better than for the Pearson correlation for larger values of threshold for several probabilistic models. In addition, we show that for some model the conditional risk for sign correlation dominates over the conditional risk for Pearson correlation for all values of threshold. These properties make sign correlation a more appropriate measure for the maximum clique analysis.

The chapter is organized as follows. In Sect. 2, we state the problem of the market graph construction and recall the concept of the true market graph. In Sect. 3, we recall the concept of the sample market graph and its relation with the true market graph. In Sect. 4, we discuss the way of measuring the accuracy of the sample market graph construction, and define the concept of conditional risk. In Sect. 5, we consider several distributions of financial asset returns that we use in our numerical experiments. In Sect. 6, we describe the results of our numerical experiments. Section 7 summarizes the results of the chapter.

## 2 Market Graph Model, True Market Graph

Let there be  $N$  financial assets observed at  $n + 1$  points in time  $t = 0, \dots, n$ .

Let  $P_i(t)$  be the price of the asset  $i$  at time point  $t$ .

$R_i(t) = \ln(P_i(t)/P_i(t-1))$  is the log-return of the asset  $i$  per period from  $t-1$  to  $t$ ,  $t = 1, \dots, n$ . We suppose that the returns  $R_i(t)$ ,  $t = 1, \dots, n$  are independent and identically distributed random variables for each fixed  $i = 1, \dots, N$ . The joint distribution of  $R_1, \dots, R_N$  determine the values of some measure of pairwise similarity. We denote by  $\Delta_{ij}$  the value of the similarity measure for  $R_i$  and  $R_j$ .

The true market graph is a simple graph constructed by the following procedure. Each of the  $N$  vertices represent a financial asset, and edges connect vertices if the measure of their pairwise similarity exceeds some threshold, that is, assets  $i$  and  $j$  are connected if and only if  $\Delta_{ij} > \Delta_0$ , where  $\Delta_0$  is the threshold. In the present chapter, we use two similarity measures producing two different true market graphs; these similarity measures are Pearson correlation and sign correlation.

Pearson correlation is the most popular measure of pairwise similarity of random variables. For two random variables  $R_i$  and  $R_j$ , it is defined as follows:

$$\rho_{i,j} = \frac{\text{cov}(R_i, R_j)}{\sigma_i \sigma_j} = \frac{E[(R_i - \mu_i)(R_j - \mu_j)]}{\sigma_i \sigma_j} \quad (1)$$

where  $\mu_i$  and  $\sigma_i$  are mean and standard deviation of  $R_i$ , respectively for  $i = 1, \dots, N$ . The values of pairwise Pearson correlation coefficients form the correlation matrix  $||\rho_{ij}||$ .

An alternative measure of similarity of random variables is the sign correlation. The application of this similarity measure to the market graph construction was proposed in [9]. The sign correlation is defined as:

$$s_{ij} = E[\text{sign}((R_i - \mu_i)(R_j - \mu_j))] \quad (2)$$

where

$$\text{sign}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

It is easy to see that sign correlation is connected to the probability of coincidence of the signs of  $(R_i - \mu_i)$  and  $(R_j - \mu_j)$ . Let  $p_{ij}$  be this probability:

$$\begin{aligned} p_{ij} &= P(R_i \geq \mu_i, R_j \geq \mu_j \vee R_i < \mu_i, R_j < \mu_j) \\ &= P(\text{sign}((R_i - \mu_i)(R_j - \mu_j)) > 0) \end{aligned}$$

The connection between probability of the coincidence of signs  $p_{ij}$  and the sign correlation looks the following way:

$$s_{ij} = 2p_{ij} - 1$$

It is easy to see that  $-1 \leq s_{ij} \leq 1$ . Interpretation of the sign correlation is quite straightforward. If  $s_{ij}$  is close to 0, the probability of sign coincidence is close 1/2,

so the variables vary independently around their mean values; if  $s_{ij}$  is close to 1, the probability of sign coincidence is close 1 as well, so the variables exceed their mean values simultaneously; similarly, if  $s_{ij}$  is close to  $-1$ , then if one of the variables exceeds its mean value, the other will most probably fall behind. For the case of a multivariate normal distribution the relationship between the sign correlation and the classic Pearson correlation is shown in [9].

Using two similarity measures, we construct two different true market graphs, which are associated with either matrix  $||\rho_{ij}||$  or matrix  $||s_{ij}||$ .

### 3 Sample Market Graph

Having a set of observations  $R_i(t)$  for  $i = 1, \dots, N$  and  $t = 1, \dots, n$ , we can estimate the values of pairwise similarities, and obtain the estimation matrix  $||\widehat{\Delta}_{ij}||$ . The sample market graph is an estimation of the true market graph, and is determined according to the statistical estimations of the measure of similarity. The sample market graph is constructed in a way similar to the true market graph construction: the sample market graph has an edge between assets  $i$  and  $j$  if and only if  $\widehat{\Delta}_{ij} > \widehat{\Delta}_0$ .

Note that the threshold value  $\widehat{\Delta}_0$  for the sample market graph can be different from the corresponding threshold value  $\Delta_0$  for the true market graph. In what follows we consider the case  $\widehat{\Delta}_0 = \Delta_0$ . This corresponds to the market graph construction procedure used in [4, 5]; the detailed analysis of this statistical procedure is given in [10].

The traditional estimator for Pearson correlation coefficient  $\rho_{ij}$  is the sample correlation:

$$\widehat{\rho}_{ij} = \frac{\sum (R_i(t) - \overline{R}_i)(R_j(t) - \overline{R}_j)}{\sqrt{\sum (R_i(t) - \overline{R}_i)^2} \sqrt{\sum (R_j(t) - \overline{R}_j)^2}} \quad (3)$$

where  $\overline{R}_i = \frac{1}{n} \sum_{t=1}^n R_i(t)$ ,  $i = 1, \dots, N$  is the estimation of the mean value. In case of multivariate normal distribution of  $R_1, \dots, R_N$ , the estimator (3) is asymptotically optimal. Consequently, in this case the sample market graph constructed on the base of the sample correlation matrix is expected to be close to the true market graph. However, if the joint distribution of  $R_1, \dots, R_N$  is not normal, this conclusion is not obvious.

For the sign correlation, the natural estimator is the sample sign correlation (also known as Fechner correlation):

$$\widehat{s}_{ij} = \frac{1}{n} \sum_{t=1}^n \text{sign}(R_i(t) - \overline{R}_i) \text{sign}(R_j(t) - \overline{R}_j) = \frac{1}{n} \sum_{t=1}^n h_{ij}(t) = \frac{e_{ij} - d_{ij}}{e_{ij} + d_{ij}} \quad (4)$$

where

$$h_{ij}(t) = \begin{cases} 1, & \text{sign}(R_i(t) - \overline{R}_i)(R_j(t) - \overline{R}_j) > 0 \\ -1, & \text{otherwise} \end{cases} \quad (5)$$

Here  $e_{ij}$  denotes the number of pairs  $\langle (R_i(t) - \overline{R_i}), (R_j(t) - \overline{R_j}) \rangle$  that have the same sign for  $t$  ( $t = 1, \dots, n$ ), and  $d_{ij}$  denotes the number of such pairs that have different signs. The estimator (4) is the number of simultaneous observations of the variables that have the same sign minus the number of simultaneous observations that have different signs, divided by the total number of observations. For financial assets, if the measure is greater than zero, the assets are likely to vary unidirectionally, if it is close to zero, they vary independently, and if the measure is less than zero, they vary in different directions.

The estimator (4) is known to be statistically robust in case of deviation of joint distribution of random variables  $R_1, \dots, R_N$  from the multivariate normal [8].

## 4 Conditional Risk

Let  $\Gamma = ||\gamma_{ij}||$  be the incidence matrix of the true market graph, where:

$$\gamma_{ij} = \begin{cases} 1, & \text{if edge between vertices } i \text{ and } j \text{ exists} \\ 0, & \text{otherwise} \end{cases}$$

Similarly we define the incidence matrix  $\widehat{\Gamma} = ||\widehat{\gamma}_{ij}||$  of the sample market graph. To measure the difference between the sample market graph and the true market graph we define the loss matrix  $L = ||l_{ij}||$  as follows:

$$l_{ij} = \begin{cases} 0, & \gamma_{ij} = \widehat{\gamma}_{ij} \\ a_{ij}, & \gamma_{ij} < \widehat{\gamma}_{ij} \\ b_{ij}, & \gamma_{ij} > \widehat{\gamma}_{ij} \end{cases}$$

When building the sample market graph, there can be two different types of errors due to inaccuracy of the similarity measure estimations:

- Some particular edge is absent in the true market graph, but it is present in the sample market graph. We denote it as error of type A, and assess the loss from it as  $a_{ij}$ .
- Some particular edge is present in the true market graph, but it is absent in the sample market graph. We denote it as error of type B, and assess the loss from it as  $b_{ij}$ .

We define the loss function of sample market graph construction as the sum of the loss matrix elements:

$$l = \sum_{i,j=1}^N l_{ij} \quad (6)$$

Note that (6) is a random variable. We define the conditional risk as its expectation:

$$R = E[l] = E \left[ \sum_{i,j=1}^N l_{ij} \right] = \sum_{i,j=1}^N E[l_{ij}] \quad (7)$$

For a fixed distribution of returns and similarity measure, the conditional risk depends only on the value of the threshold, therefore we consider it as a function of  $\Delta_0$ :  $R = R(\Delta_0)$ .

If all financial assets are considered to be equally important, we can assume that  $a_{ij} = a$  and  $b_{ij} = b$ , for all  $i, j = 1, \dots, N$ . For simplicity, we take  $a + b = 1$ . In this case, the conditional risk is:

$$R(\Delta_0) = am_a(\Delta_0) + bm_b(\Delta_0)$$

where  $m_a(\Delta_0)$  is the expectation of number of errors of type A and  $m_b(\Delta_0)$  is the expectation of number of errors of type B.

In this chapter, we consider the case  $a = b = 1/2$ , and use the Monte-Carlo simulation to study the conditional risk for two measures of similarity: Pearson correlation and sign correlation. In the next section, we describe the data generator.

## 5 Return Distribution Generator

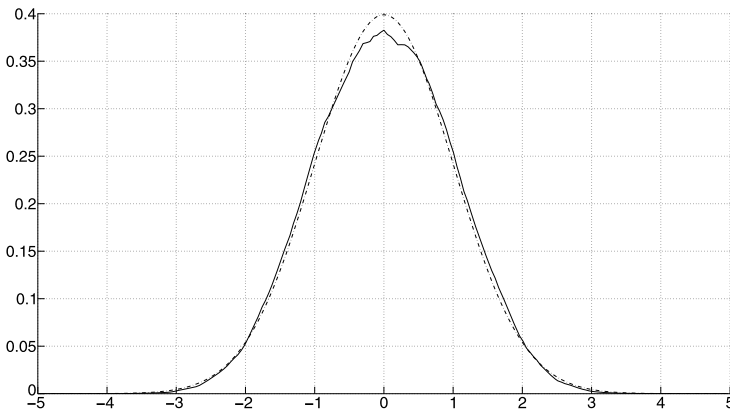
We construct the data generator with fixed Pearson correlation matrix and different return distributions. To study the behavior of the conditional risk in case of deviation from normal distribution, we use the following generator. Let  $X_1, \dots, X_N$  be independent identically distributed variables with a given generating distribution,  $E(X_i) = 0$ ,  $\sigma^2(X_i) = 1$  for  $i = 1, \dots, N$ . Let  $\Sigma$  be the fixed correlation matrix. We define the resulting vector of random variables as:

$$R = \Sigma^{1/2} X \tag{8}$$

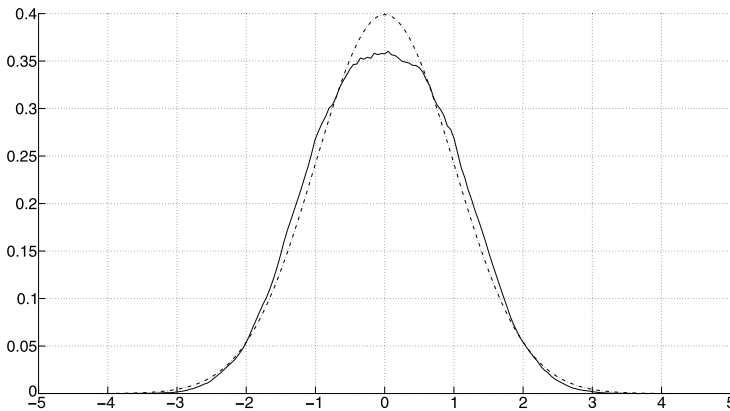
The correlation matrix of constructed this way random variables exactly equals to  $\Sigma$ . To obtain  $R_i(t)$ , we generate a set of values  $X_i(t)$  for  $t = 1, \dots, n$  and calculate  $R_i(t)$  by (8). Random variables  $R_i(t)$  are independent and identically distributed for a fixed  $i$ . The marginal distributions of  $R_1, \dots, R_N$  depend on the generating distribution of  $X_1, \dots, X_N$  and the correlation matrix  $\Sigma$ .

We consider the following multivariate distributions of the financial asset returns:

- Let  $R_1, \dots, R_N$  have a multivariate normal distribution with zero means and correlation matrix  $\Sigma$ . Multivariate normal is one of the most commonly used distributions for the financial market analysis.
- Let  $R_1, \dots, R_N$  have a multivariate t-distribution with 3 degrees of freedom and the correlation matrix  $\Sigma$ . Multivariate t-distribution is also frequently used for modeling of the financial asset returns. It is interesting to consider the distribution with a little number of degrees of freedom, because in this case t-distribution is substantially different from normal.
- Let variables  $X_1, \dots, X_N$  be distributed uniformly on the interval from  $-\sqrt{3}$  to  $\sqrt{3}$ . In this case, for a real market correlation matrix  $\Sigma$ , the distributions of  $R_1, \dots, R_N$  obtained with (8) are platykurtic. Figure 1 shows a typical marginal distribution of  $R_i$ .



**Fig. 1** A typical marginal distribution of  $R_i$  for the uniform generating distribution of  $X_1, \dots, X_N$ . The *solid line* is the distribution density, the *dashed line* is the density of normal distribution for comparison



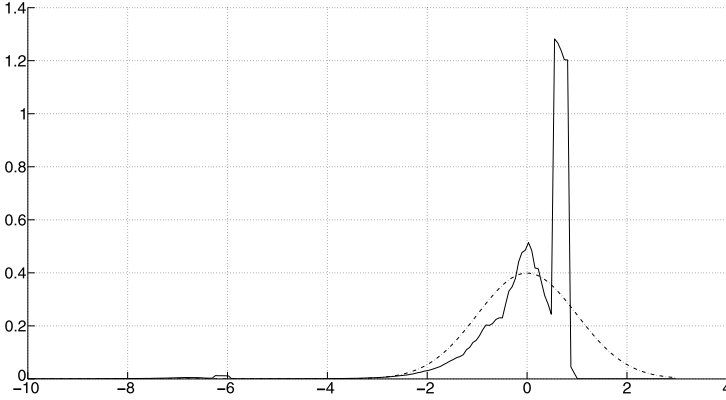
**Fig. 2** A typical marginal distribution of  $R_i$  for the bimodal generating distribution of  $X_1, \dots, X_N$ . The *solid line* is the distribution density, the *dashed line* is the density of normal distribution for comparison

- The distribution density of  $X_1, \dots, X_N$  is given by:

$$f(x) = \begin{cases} ax^2, & |x| < b \\ 0, & |x| \geq b \end{cases}$$

where  $a = \frac{9}{10}\sqrt{\frac{3}{5}}$  and  $b = \sqrt{\frac{5}{3}}$ . The generating distribution density has two distinct peaks, so we denote it as bimodal distribution. Figure 2 shows a typical marginal distribution of  $R_i$  obtained by (8).

- To emphasize the peculiarity of sign correlation as the measure of pairwise similarity we use a specific distribution of the following type. It models the situation



**Fig. 3** A typical marginal distribution of  $R_i$  for the distribution modeling stable moderate positive trend alternating with rare but significant downturns. The *solid line* is the distribution density, the *dashed line* is the density of normal distribution for comparison

of stable moderate positive trend alternating with rare but significant downturns. The variables  $X_1, \dots, X_N$  take values  $-10$  and  $0.1$  with probabilities  $1/101$  and  $100/101$ , respectively. Figure 3 shows a typical marginal distribution of  $R_i$  obtained by (8).

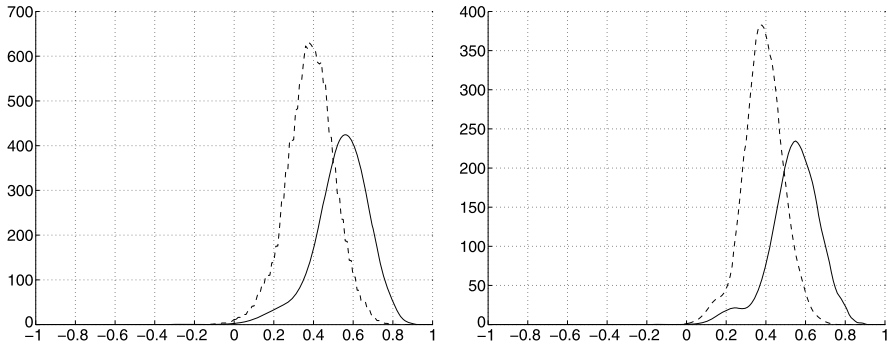
- We also consider a truncated multivariate normal distribution. The range of random variables variance is limited to interval  $[-\theta; +\theta]$ . This model simulates the practice of stopping operations on the financial market in case of significant changes during a period. We consider  $\theta = 0.1$ , that is, the observations are truncated to the range of  $[-0.1; 0.1]$ . Note that the correlation matrix of the resulting random variables is different from  $\Sigma$ . Therefore, for constructing of the Pearson-based real market graph we additionally calculate the actual values of correlations between the resulting random variables.

## 6 Numerical Experiments

Our numerical experiments have the following structure. We adopt some assumption about the joint distribution of  $R_1, \dots, R_N$ . Then, we create a set of pseudo-random observations according to the adopted model, and construct the sample market graph. The experiment is repeated many times (in our case—1000 times), then the average numbers of errors of type A and errors of type B, as well as the average value of the conditional risk are calculated. The resulting values depend on the initially adopted distribution, the threshold value  $\Delta_0$ , and the number of generated observations  $n$ .

We take the correlation matrix of the assets in the US financial market as the initial correlation matrix  $\Sigma = ||\sigma_{ij}||$ , the number of assets is  $N = 100$ . The true sign correlation matrix is estimated separately for each considered distribution model by





**Fig. 4** Conditional risk of the market graph construction  $R(\Delta_0)$ ,  $-1 \leq \Delta_0 \leq 1$  for the multivariate normal distribution. Number of observations  $n = 100$  (left) and  $n = 400$  (right). The solid line is the conditional risk for the classic correlation, and the dashed line is the conditional risk for the sign correlation

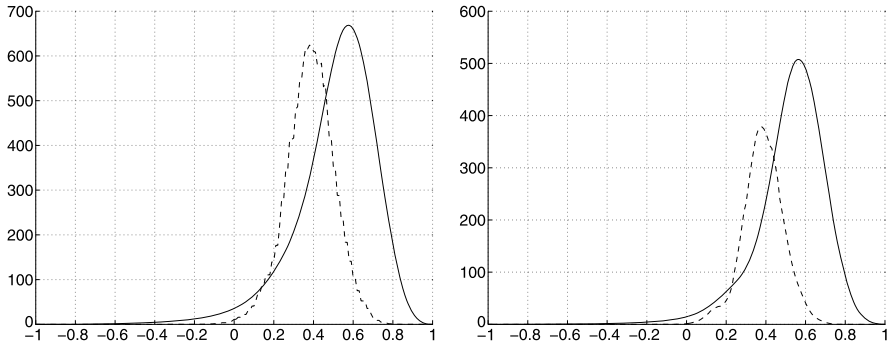
generation of 100,000 observations and direct calculation. For the truncated multivariate normal model, the true values of the classic correlations are also estimated using a large number of generated observations. For each distribution of asset returns, for a given threshold  $\Delta_0$ , we have two true and two sample market graphs: based on the classic correlation, and based on the sign correlation.

For each model, we build a graph of the conditional risk depending on the threshold value  $\theta_0$ . It is done for the two methods of the market graph construction (classic correlation and sign correlation), and for two values of the number of observations  $n = 100$  and  $n = 400$ .

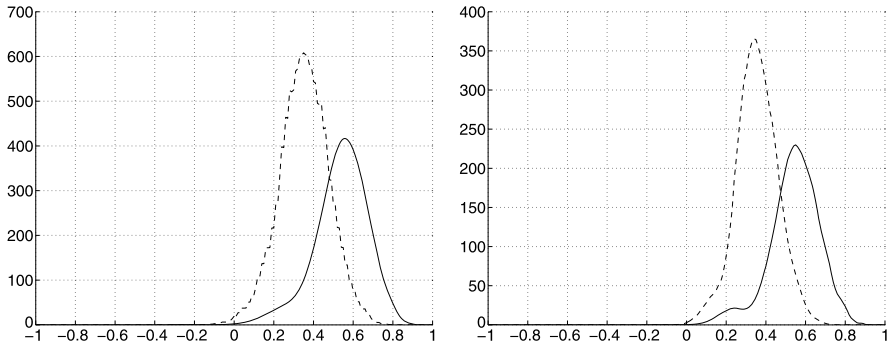
The calculations are running on a machine equipped with Intel(R) Core(TM) i3 processor and 8 gigabytes of RAM with Ubuntu 12.04 operating system using Matlab R2011b. Calculation of one characteristic for a fixed  $\Delta_0$  takes in average 0.3 seconds of CPU Time.

Figure 4 shows the graphs of the conditional risk  $R(\Delta_0)$ ,  $-1 \leq \Delta_0 \leq 1$  for the multivariate normal distribution with number of observations  $n = 100$  and  $n = 400$ . We can conclude that for  $\Delta_0 > 0.5$  the conditional risk for the sign correlation is lower, that is, for larger thresholds values the market graph based on the sign correlation can be estimated more accurately.

Figure 5 shows the graphs of the conditional risk for the multivariate t-distribution model with 3 degrees of freedom. Although these graphs resemble the graphs for the multivariate normal model, a substantial difference is that the market graph based on the classic correlation coefficient turns out to be better in the sense of conditional risk only for a limited range of threshold values. It means, that for the probabilistic models similar to the multivariate t-distribution, the sign correlation is more suitable not only for large thresholds (which are important for the maximum clique analysis), but also for low thresholds (which are important for the maximum independent set analysis). In addition, the maximum value of conditional risk for the sign correlation is lower than the maximum value of conditional risk for the classic correlation.



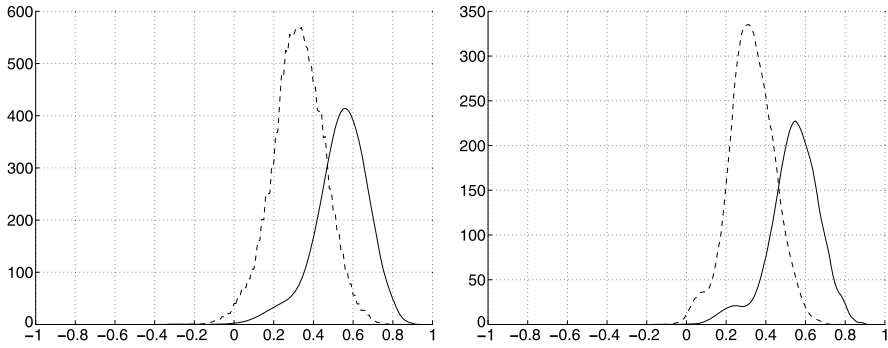
**Fig. 5** Conditional risk of the market graph construction  $R(\Delta_0)$ ,  $-1 \leq \Delta_0 \leq 1$  for the multivariate t-distribution with 3 degrees of freedom. Number of observations  $n = 100$  (left) and  $n = 400$  (right). The *solid line* is the conditional risk for the classic correlation, and the *dashed line* is the conditional risk for the sign correlation



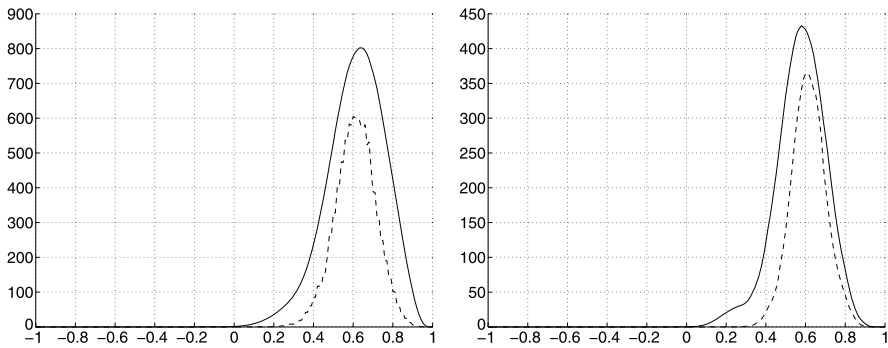
**Fig. 6** Conditional risk of the market graph construction  $R(\Delta_0)$ ,  $-1 \leq \Delta_0 \leq 1$  for the platykurtic distribution. Number of observations  $n = 100$  (left) and  $n = 400$  (right). The *solid line* is the conditional risk for the classic correlation, and the *dashed line* is the conditional risk for the sign correlation

Figures 6 and 7 show the graphs of  $R(\Delta_0)$  for platykurtic and bimodal distributions (see the previous section for the description of the distributions). Generally they look very similar to each other and to the graphs for the normal model, although the generating distributions are different.

The peculiarity of the sign correlation as a measure of pairwise similarity is perfectly emphasized by the model with moderate positive trend alternating with rare but significant downturns. The graphs for this model are shown in Fig. 8. We can see that for a distribution of this kind the market graph based on the sign correlation model is constructed more accurately than the market graph based on the classic Pearson correlation. For the number of observations  $n = 100$  and  $n = 400$ , the whole graphs for the sign correlation lie below the graphs for the classic correlation, that



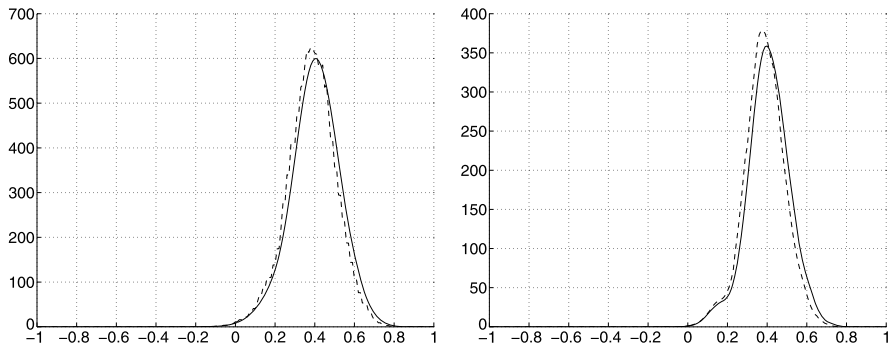
**Fig. 7** Conditional risk of the market graph construction  $R(\Delta_0)$ ,  $-1 \leq \Delta_0 \leq 1$  for the bimodal distribution. Number of observations  $n = 100$  (left) and  $n = 400$  (right). The *solid line* is the conditional risk for the classic correlation, and the *dashed line* is the conditional risk for the sign correlation



**Fig. 8** Conditional risk of the market graph construction  $R(\Delta_0)$ ,  $-1 \leq \Delta_0 \leq 1$  for the distribution modeling stable trend and rare risk. Number of observations  $n = 100$  (left) and  $n = 400$  (right). The *solid line* is the conditional risk for the classic correlation, and the *dashed line* is the conditional risk for the sign correlation

is, the loss of the sample market graph construction is less for the sign correlation based graph for all the range of the thresholds  $-1 \leq \Delta_0 \leq 1$ .

Figure 9 shows the graphs of the conditional risk for the truncated multivariate normal model. For this model, we truncate the normal at range  $[-0.1; 0.1]$ , as a result most of the observations take the values of either  $-0.1$  or  $0.1$ . The features of the classic correlation coefficient for such a distribution become similar to the features of the sign correlation, and it is fully confirmed by the graphs. The conditional risks for the two market graphs are close to each other for all values of the threshold  $\Delta_0$ .



**Fig. 9** Conditional risk of the market graph construction  $R(\Delta_0)$ ,  $-1 \leq \Delta_0 \leq 1$  for the multivariate normal distribution truncated at range  $[-0.1; 0.1]$ . Number of observations  $n = 100$  (left) and  $n = 400$  (right). The *solid line* is the conditional risk for the classic correlation, and the *dashed line* is the conditional risk for the sign correlation

## 7 Conclusions

We studied the conditional risk of the market graph construction. It was shown that for return distributions different from multivariate normal, the market graph based on the sign correlation can be constructed with a better accuracy than the market graph based on the classic correlation. Despite of the common use of the Pearson correlation for the market network analysis, there are the cases where a different measure of similarity is preferable. The comparison of the market graph analysis based on the Pearson correlation and the sign correlation shows:

- For a large class of return distributions there exists such a threshold value  $\Delta_0^*$ , that for all  $\Delta_0 \geq \Delta_0^*$ , the construction of the market graph based on the sign correlation can be done much more precisely. This makes the sign correlation a preferable measure of similarity for the maximum clique analysis [4–6].
- For the values of threshold  $\Delta_0 < \Delta_0^*$ , Pearson correlation is better as a measure similarity. This is crucial for the maximum independent set analysis.
- In some cases, the sign correlation completely dominates the Pearson correlation in the sense of conditional risk. In particular, this is the case of the distribution modeling a market situation with a stable positive trend alternating with rare but significant downturns.
- For distributions with heavy tails, the maximum value of conditional risk for the sign correlation is less than the maximum value of conditional risk for the Pearson correlation. Therefore, the sign correlation is a preferable similarity measure for minmax criteria for such distributions [8].

**Acknowledgements** The authors are partially supported by LATNA Laboratory, NRU HSE, RF government grant, ag. 11.G34.31.0057.

## References

1. Mantegna, R.N.: Hierarchical structure in financial market. *Eur. Phys. J. B* **11**, 193–197 (1999)
2. Tumminello, M., Aste, T., Di Matteo, T., Mantegna, R.N.: *Proc. Natl. Acad. Sci. USA* **102**(30), 10421–10426 (2005)
3. Tumminello, M., Lillo, F., Mantegna, R.N.: Correlation, hierarchies and networks in financial markets. *J. Econ. Behav. Organ.* **75**, 40–58 (2010)
4. Boginsky, V., Butenko, S., Pardalos, P.M.: On structural properties of the market graph. In: Nagurney, A. (ed.) *Innovations in Financial and Economic Networks*, pp. 29–45. Edward Elgar, Northampton (2003)
5. Boginsky, V., Butenko, S., Pardalos, P.M.: Statistical analysis of financial networks. *Comput. Stat. Data Anal.* **48**, 431–443 (2005)
6. Boginsky, V., Butenko, S., Pardalos, P.M.: Mining market data: a network approach. *Comput. Oper. Res.* **33**, 3171–3184 (2006)
7. Anderson, T.W.: *An Introduction to Multivariate Statistical Analysis*, 3rd edn. Wiley-Interscience, New York (2003)
8. Stuart, A., Ord, J.K., Arnold, S.: *Kendall's Advanced Theory of Statistics*, vol. 2A, Classical Inference and the Linear Model, 6th edn. Wiley, New York (2004)
9. Bautin, G.A., Kalyagin, V.A., Koldanov, A.P., Koldanov, P.A., Pardalos, P.M.: Simple measure of similarity for the market graph construction. *Comput. Manag. Sci.* (2013). doi:[10.1007/s10287-013-0169-3](https://doi.org/10.1007/s10287-013-0169-3)
10. Koldanov, A.P., Koldanov, P.A., Kalyagin, V.A., Pardalos, P.M.: Statistical procedures for the market graph construction. *Comput. Stat. Data Anal.* (2013, to appear)

# Heuristic Algorithm for the Cell Formation Problem

Ilya Bychkov, Mikhail Batsyn, Pavel Sukhov, and Panos M. Pardalos

**Abstract** In this chapter, we introduce a new heuristic for Cell Formation Problem in its most general formulation with grouping efficiency as an objective function. Suggested approach applies an improvement procedure to obtain solutions with high grouping efficiency. This procedure is repeated until efficiency can be increased for randomly generated configurations of cells. We consider our preliminary results for 10 popular benchmark instances taken from the literature. Also source instances with the solutions we got can be found in the [Appendix](#).

**Keywords** Cell formation · Grouping efficiency · Improvement heuristic

## 1 Introduction

Flanders (1925) [7] was the first who formulated the main ideas of the group technology. The notion of the Group Technology was introduced in Russia by Mitrofanov (1959) [9], though his work was translated to English only in 1966 (Mitrofanov, 1966 [10]). One of the main problems stated by the Group Technology is the optimal formation of manufacturing cells, that is, grouping of machines and parts

---

I. Bychkov (✉) · M. Batsyn · P. Sukhov · P.M. Pardalos

Laboratory of Algorithms and Technologies for Network Analysis, National Research University Higher School of Economics, 136 Rodionova Street, Nizhny Novgorod, 603093, Russian Federation

e-mail: [il.bychkov@gmail.com](mailto:il.bychkov@gmail.com)

M. Batsyn

e-mail: [mbatsyn@hse.ru](mailto:mbatsyn@hse.ru)

P. Sukhov

e-mail: [pavelandreevith@rambler.ru](mailto:pavelandreevith@rambler.ru)

P.M. Pardalos

e-mail: [pardalos@ufl.edu](mailto:pardalos@ufl.edu)

P.M. Pardalos

Center of Applied Optimization, University of Florida, 401 Weil Hall, P.O. Box 116595, Gainesville, FL 32611-6595, USA

into cells such that for every machine in a cell the number of the parts from this cell processed by this machine is maximized and the number of the parts from other cells processed by this machine is minimized. In other words, the intra-cell loading of machines is maximized and simultaneously the inter-cell movement of parts is minimized. This problem is called the Cell Formation Problem (CFP). Burbidge (1961) [3] suggested his Product Flow Analysis (PFA) approach for the CFP, and later popularized the Group Technology and the CFP in his book (Burbidge, 1975 [4]).

The CFP is NP-hard since it can be reduced to the clustering problem (Ghosh et al., 1996 [1]). That is why there is a great number of heuristic approaches for solving CFP and almost no exact ones. Our heuristic algorithm is based on sequential improvements of the solution. We modify the cell configuration by enlarging one cell and reducing another. The basic procedure of the algorithm has the following steps:

1. Generate a random cell configuration.
2. Improve the initial solution moving one row or column from one cell to another until the grouping efficiency is increasing.
3. Repeat steps 1–2 a predefined number of times (we use 2000 times for computational experiments in this chapter).

The chapter is organized as follows. In the next section, we provide the Cell Formation Problem formulation. In Sect. 3, we present our improvement heuristic that allows us to get good solutions by iterative modifications of cells which lead to increasing of the objective function. In Sect. 4, we report our computational results and Sect. 5 concludes the chapter with a short summary.

## 2 The Cell Formation Problem

The CFP consists in an optimal grouping of the given machines and parts into cells. The input for this problem is given by  $m$  machines,  $p$  parts and a rectangular machine-part incidence matrix  $A = [a_{ij}]$ , where  $a_{ij} = 1$  if part  $j$  is processed on machine  $i$ . The objective is to find an optimal number and configuration of rectangular cells (diagonal blocks in the machine-part matrix) and optimal grouping of rows (machines) and columns (parts) into these cells such that the number of zeros inside the chosen cells (voids) and the number of ones outside these cells (exceptions) are minimized. A concrete combination of rectangular cells in a solution (diagonal blocks in the machine-part matrix) we will call a cells configuration.

For example, we are given the machine-part matrix (Waghodekar and Sahu, 1984 [13]) shown in Table 1. A feasible solution for this CFP is shown in Table 2.

There are a number of different objective functions used for the CFP. For our research we used so called grouping efficiency measure suggested by Chandrasekharan and Rajagopalan (1989) [6] and compare our computational results with the

**Table 1** Machine-part  $5 \times 7$  matrix from Waghodekar and Sahu (1984) [13]

	$p_1$	$p_2$	$p_3$	$p_4$	$p_5$	$p_6$	$p_7$
$m_1$	1	0	0	0	1	1	1
$m_2$	0	1	1	1	1	0	0
$m_3$	0	0	1	1	1	1	0
$m_4$	1	1	1	1	0	0	0
$m_5$	0	1	0	1	1	1	0

**Table 2** CFP solution

	$p_7$	$p_6$	$p_1$	$p_5$	$p_3$	$p_2$	$p_4$
$m_1$	1	1	1	1	0	0	0
$m_4$	0	0	1	0	1	1	1
$m_3$	0	1	0	1	1	0	1
$m_2$	0	0	0	1	1	1	1
$m_5$	0	1	0	1	0	1	1

results of Bhatnagar and Saddikuti (2010) [16] and Goldengorin et al. (2012) [8]. Grouping efficiency described by Chandrasekharan and Rajagopalan (1989) [6] by the following formula:

$$\eta = q\eta_1 + (1 - q)\eta_2, \quad (1)$$

where

$$\eta_1 = \frac{n_1 - n_1^{\text{out}}}{n_1 - n_1^{\text{out}} + n_0^{\text{in}}} = \frac{n_1^{\text{in}}}{n^{\text{in}}}$$

$$\eta_2 = \frac{mp - n_1 - n_0^{\text{in}}}{mp - n_1 - n_0^{\text{in}} + n_1^{\text{out}}} = \frac{n_0^{\text{out}}}{n^{\text{out}}}$$

- $\eta_1$ —a ratio showing the intra-cell loading of machines (or the ratio of the number of ones in cells to the total number of elements in cells).
- $\eta_2$ —a ratio inverse to the inter-cell movement of parts (or the ratio of the number of zeroes out of cells to the total number of elements out of cells).
- $q$ —a coefficient ( $0 \leq q \leq 1$ ) reflecting the weights of the machine loading and the inter-cell movement in the objective function. It is usually taken equal to  $\frac{1}{2}$ , which means that it is equally important to maximize the machine loading and minimize the inter-cell movement.
- $n_1$ —a number of ones in the machine-part matrix,
- $n_0$ —a number of zeroes in the machine-part matrix,
- $n^{\text{in}}$ —a number of elements inside the cells,
- $n^{\text{out}}$ —a number of elements outside the cells,
- $n_1^{\text{in}}$ —a number of ones inside the cells,



- $n_1^{\text{out}}$ —a number of ones outside the cells,
- $n_0^{\text{in}}$ —a number of zeroes inside the cells,
- $n_0^{\text{out}}$ —a number of zeroes outside the cells.

Efficiency values the solution in Table 2 is calculated below.

$$\eta = \frac{1}{2} \cdot \frac{16}{19} + \frac{1}{2} \cdot \frac{12}{16} \approx 79.60 \%$$

The mathematical programming model of the CFP with the grouping efficiency objective function can be described using boolean variables  $x_{ik}$  and  $y_{jk}$ . Variable  $x_{ik}$  takes value 1 if machine  $i$  belongs to cell  $k$  and takes value 0 otherwise. Similarly variable  $y_{jk}$  takes value 1 if part  $j$  belongs to cell  $k$  and takes value 0 otherwise. Machines index  $i$  takes values from 1 to  $m$  and parts index  $j$ —from 1 to  $p$ . Cells index  $k$  takes values from 1 to  $c = \min(m, p)$  because every cell should contain at least one machine and one part and so the number of cells cannot be greater than  $m$  and  $p$ . Note that if a CFP solution has  $n$  cells then for  $k$  from  $n + 1$  to  $c$  all variables  $x_{ik}, y_{jk}$  will be zero in this model. So we can consider that the CFP solution always has  $c$  cells, but some of them can be empty. The mathematical programming formulation is as follows.

$$\max \left( \frac{n_1^{\text{in}}}{2n^{\text{in}}} + \frac{n_0^{\text{out}}}{2n^{\text{out}}} \right) \quad (2)$$

where

$$\begin{aligned} n^{\text{in}} &= \sum_{k=1}^c \sum_{i=1}^m \sum_{j=1}^p x_{ik} y_{jk}, & n^{\text{out}} &= mp - n^{\text{in}} \\ n_1^{\text{in}} &= \sum_{k=1}^c \sum_{i=1}^m \sum_{j=1}^p a_{ij} x_{ik} y_{jk}, & n_0^{\text{out}} &= n_0 - (n^{\text{in}} - n_1^{\text{in}}) \end{aligned}$$

subject to

$$\sum_{k=1}^c x_{ik} = 1 \quad \forall i \in 1, \dots, m \quad (3)$$

$$\sum_{k=1}^c y_{jk} = 1 \quad \forall j \in 1, \dots, p \quad (4)$$

$$\sum_{i=1}^m \sum_{j=1}^p x_{ik} y_{jk} \geq \sum_{i=1}^m x_{ik} \quad \forall k \in 1, \dots, c \quad (5)$$

$$\sum_{i=1}^m \sum_{j=1}^p x_{ik} y_{jk} \geq \sum_{j=1}^p y_{jk} \quad \forall k \in 1, \dots, c \quad (6)$$

$$x_{ik} \in \{0, 1\} \quad \forall i \in 1, \dots, m \quad (7)$$

$$y_{jk} \in \{0, 1\} \quad \forall j \in 1, \dots, p \quad (8)$$

The objective function (5) is the grouping efficiency in this model. Constraints (6) and (7) impose that every machine and every part belongs to some cell. Constraints (8) and (9) guarantee that every non-empty cell contains at least one machine and one part. Note that if singleton cells are not allowed then the right sides of inequalities (8) and (9) should have a coefficient of 2. All these constraints can be linearized in a standard way, but the objective function will still be fractional. That is why the exact solution of this problem presents considerable difficulties.

### 3 Algorithm Description

Our algorithm is as follows.

1. First, we look for a potentially optimal range of cells.
  - a. Look over all the possible number of cells from 2 to maximal possible number of cells which is equal to  $\min(m, p)$ .
  - b. For every number of cells in this interval, we generate a fixed number of configurations (we use 500 in this chapter).
  - c. For every of these 500 configurations, we try to rearrange rows and columns between cells in order to increase efficiency value. These rearrangements are made until efficiency could be increased.
  - d. The best efficiency value from 500 results above are chosen. This would be our best bound for this particular number of cells.
  - e. Then we get the best efficiency value from bounds for cells number from 2 to  $\min(m, p)$  and its 10 percent neighborhood is optimal cell range in that our following procedures will be made.
2. For every number of cells in our optimal interval, we do a more precise search.
  - a. For every number of cells in the optimal interval, we generate a larger number of configurations (we use 2000 in this chapter).
  - b. For every of these 2000 configurations, we try to rearrange rows and columns between cells in order to increase efficiency value. These rearrangements are made until efficiency could be increased.
  - c. The best efficiency value from these 2000 results above are chosen. This would be our best bound for this particular number of cells.
  - d. Then we get the best efficiency value for our optimal cell range and this result is our final for this problem instance.

The main idea of the improvement procedure (that rearranges rows and columns) is illustrated on Seifoddini and Wolfe (1986) [12] instance  $8 \times 12$  (Table 3). To compute the grouping efficiency for this solution, we need to know the number of ones inside cells  $n_1^{\text{in}}$ , the total number of elements inside cells  $n^{\text{in}}$ , the number of

**Table 3** Seifoddini and Wolfe (1986) [12] instance  $8 \times 12$ 

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	0	0	0	0	0	0	0	0
2	1	0	1	1	1	1	1	0	0	1	0	0
3	0	0	1	1	1	1	1	1	1	0	0	0
4	0	0	0	0	0	1	1	1	1	1	0	0
5	0	0	0	0	0	0	1	1	1	1	0	0
6	0	0	0	0	0	0	1	1	1	0	1	0
7	0	0	0	0	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	0	0	0	0	1	1

**Table 4** Moving part 4 from cell 2 to cell 1

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	0	0	0	0	0	0	0	0
2	1	0	1	1	1	1	1	0	0	1	0	0
3	0	0	1	1	1	1	1	1	1	0	0	0
4	0	0	0	0	0	1	1	1	1	1	0	0
5	0	0	0	0	0	0	1	1	1	1	0	0
6	0	0	0	0	0	0	1	1	1	0	1	0
7	0	0	0	0	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	0	0	0	0	1	1

zeros outside cells  $n_0^{\text{out}}$  and the number of elements outside cells  $n^{\text{out}}$ . The grouping efficiency is then calculated by the following formula:

$$\eta = q \cdot \frac{n_1^{\text{in}}}{n^{\text{in}}} + (1 - q) \cdot \frac{n_0^{\text{out}}}{n^{\text{out}}} = \frac{1}{2} \cdot \frac{20}{33} + \frac{1}{2} \cdot \frac{48}{63} \approx 68.4 \%$$

Looking at this solution (Table 3), we can conclude that it is possible for example, to move part 4 from the second cell to the first one. And this way the number of zeros inside cells decreases by 3 and the number of ones outside cells also decreases by 4. So it is profitable to attach column 4 to the first cell as it is shown on Table 4.

For the modified cells configuration we have:

$$\eta = \frac{1}{2} \cdot \frac{23}{33} + \frac{1}{2} \cdot \frac{51}{63} \approx 75.32 \%$$

As a result, the efficiency is increased almost for 7 percent. Computational results show that using such modifications could considerably improve the solution. The idea is to compute an increase in efficiency for each column and row when it is moved to another cell and then perform the modification corresponding to the maximal increase. For example, Table 5 shows the maximal possible increase in efficiency for every row when it is moved to another cell.

**Table 5** Maximal efficiency increase for each row

	1	2	3	4	5	6	7	8	9	10	11	12	
1	1	1	1	1	0	0	0	0	0	0	0	0	−6.94 %
2	1	0	1	1	1	1	1	0	0	1	0	0	+1.32 %
3	0	0	1	1	1	1	1	1	1	0	0	0	+7.99 %
4	0	0	0	0	0	1	1	1	1	1	0	0	−0.07 %
5	0	0	0	0	0	0	1	1	1	1	0	0	+0.77 %
6	0	0	0	0	0	0	1	1	1	0	1	0	+0.77 %
7	0	0	0	0	0	0	0	0	0	0	1	1	−4.62 %
8	0	0	0	0	0	0	0	0	0	0	1	1	−4.62 %

4 Computational Results

In all the experiments for determining a potentially optimal range of cells, we use 500 random cell configurations for each cells number and for obtaining the final solution we use 2000 random configurations. An Intel Core i7 machine with 2.20 GHz CPU and 8.00 Gb of memory is used in our experiments. We run our heuristic on 10 CFP benchmark instances taken from the literature. The computational results are presented in Table 6. For every instance, we make 50 algorithm runs and report minimum, average and maximum value of the grouping efficiency obtained by the suggested heuristic over these 50 runs. We compare our results with the best known values taken from Goldengorin et al. (2012) [8] and Bhatnagar and Saddikuti (2010) [16]. Our final solutions are presented in Tables 7–26.

5 Concluding Remarks

In this chapter, we present a new heuristic algorithm for solving the CFP. The high quality of the solutions is achieved due to the enumeration of different numbers of cells and different cell configurations and applying our improvement procedure. Since the suggested heuristic works fast, we apply it for thousands of different configurations. Thus, a big variety of good solutions is covered by the algorithm and the best of them has high grouping efficiency.

**Acknowledgements** The authors are partially supported by LATNA Laboratory, NRU HSE, RF government grant, ag. 11.G34.31.0057.

Table 6 Computational results

#	Source	$m \times p$	Efficiency value, %		Time, sec		
			Bhatnagar and Saddikuti	Goldengorin et al.	Our		
					Min	Avg	Max
1	Sandbothe (1998) [14]	$10 \times 20$	95.40	95.93	95.66	95.66	95.66
2	Ahi et al. (2009) [15]	$20 \times 20$	92.62	93.85	95.99	95.99	95.99
3	Mosier and Taube (1985) [11]	$20 \times 20$	85.63	88.71	90.11	90.16	90.22
4	Boe and Cheng (1991) [2]	$20 \times 35$	88.31	88.05	93.34	93.47	93.55
5	Carrie (1973) [5]	$20 \times 35$	90.76	95.64	95.43	95.78	95.79
6	Ahi et al. (2009) [15]	$20 \times 51$	87.86	94.11	95.36	95.4	95.45
7	Chandrasekharan and Rajagopalan (1989) [6]	$24 \times 40$	98.82	100.00	100	100	100
8	Chandrasekharan and Rajagopalan (1989) [6]	$24 \times 40$	95.33	97.48	97.7	97.75	97.76
9	Chandrasekharan and Rajagopalan (1989) [6]	$24 \times 40$	93.78	96.36	96.84	96.88	96.89
10	Chandrasekharan and Rajagopalan (1989) [6]	$24 \times 40$	87.92	94.32	96.11	96.16	96.21

**Table 7** Sandbothe (1998) [14]**Table 8** Ahi et al. (2009) [15][illegible]



**Table 10** Boe and Cheng (1991) [2] 35 parts and 20 machines

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0
2	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0
3	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0
5	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0
6	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0
7	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
8	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	0
10	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
11	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0
12	0	1	0	1	0	0	1	0	0	0	0	0	1	1	0	1	0	1	0	0
13	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0
14	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1
15	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0
16	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
17	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0
18	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
19	0	1	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	1
20	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0
21	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0
22	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
23	1	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1
24	1	1	0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0	0
25	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
26	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1
27	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
29	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
30	1	0	0	0	0	1	1	0	0	0	1	0	0	0	1	1	0	0	1	0
31	0	1	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
32	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1	0
33	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
34	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	1	0	0	0





**Table 12** Ahi et al. (2009) [15] 51 parts and 20 machines

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
6	1	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
7	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0
10	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0
11	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1
12	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0
15	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0
16	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1
17	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0
18	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0
19	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1
20	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
21	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
22	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0
23	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0
24	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0
25	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0
26	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1
27	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1
28	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
29	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
30	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0
31	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0
32	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1
33	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
34	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
35	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
36	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0
37	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0
38	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1
39	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	0
40	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	1	0
41	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1
42	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0
43	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0
44	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1
45	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0
46	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0
47	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1
48	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0
49	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
50	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
51	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

**Table 13** Chandrasekharan and Rajagopalan (1989) [6] 40 parts and 24 machines

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
10	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
12	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
13	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
14	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
15	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
16	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
17	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
18	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
19	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
21	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
22	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
23	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
24	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
25	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1
26	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
27	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
28	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
30	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
31	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
32	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1
33	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
34	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
35	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
36	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
37	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
38	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
39	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

**Table 14** Chandrasekharan and Rajagopalan (1989) [6] 40 parts and 24 machines

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
3	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
5	0	0	0	0	1	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
10	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
12	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
13	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
14	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
15	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
16	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
17	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
18	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
19	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0
21	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
22	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
23	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
24	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
25	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0
26	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0
27	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
28	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
30	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
31	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
32	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1
33	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
34	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
35	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
36	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
38	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
39	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

**Table 15** Chandrasekharan and Rajagopalan (1989) [6] 40 parts and 24 machines

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
5	0	0	0	0	1	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
10	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
12	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
13	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
14	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
15	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
16	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0
17	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
18	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
19	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
20	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0
21	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
22	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
23	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
24	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
25	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0
26	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0
27	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
28	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
30	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
32	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
33	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
34	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
35	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
36	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
38	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
39	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

**Table 16** Chandrasekharan and Rajagopalan (1989) [6] 40 parts and 24 machines

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	1
4	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
5	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
11	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
12	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
13	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0
14	1	1	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0
15	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0
17	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0
18	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
19	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
20	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0
21	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
22	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
24	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
26	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	0
28	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
30	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
32	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
33	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
34	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
35	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0	1	0	0	0	0	0
36	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
38	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

**Table 17** Sandbothe (1998) [14] solution

	1	4	3	6	2	5	12	13	11	9	14	8	7	15	17	16	18	10	19	20
1	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	<b>1</b>	1	1	1	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	<b>1</b>	0	1	1	0	0	0	0	1	0	0
5	0	0	0	0	0	0	1	0	1	<b>1</b>	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	1	<b>1</b>	<b>1</b>	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	1	0	1	<b>1</b>	<b>1</b>	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>

**Table 18** Ahi et al. (2009) [15] solution

	8	7	6	5	16	13	14	10	3	18	17	12	1	9	20	15	11	2	4	19
13	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
4	<b>1</b>	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
17	0	<b>1</b>	<b>1</b>	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
16	0	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
15	1	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	<b>1</b>	0	0	1	0	0	0	0	1	0	0	0	1	0
19	0	0	0	0	1	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	1	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	1	1	1	<b>1</b>	0	0	0	0	0
14	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	<b>1</b>	<b>1</b>	<b>1</b>
2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	<b>1</b>	<b>1</b>	<b>1</b>

[illegible]





**Table 21** Carrie (1973) [5] 35 parts and 20 machines solution

	15	16	19	11	12	1	7	20	17	3	8	6	9	10	14	2	18	4	13	5
11	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	1
22	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
14	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	1
19	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
31	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
16	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1
34	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

**Table 22** Ahi et al. (2009) [15] 51 parts and 20 machines solution

	14	10	17	4	9	20	15	13	7	2	16	1	12	19	5	6	8	11	3	18
39	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
41	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
48	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
51	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
44	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
47	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0	0	0	0	0
49	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	1
13	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	0
26	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0
28	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0
25	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	1
21	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0
22	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0
20	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0
11	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0
10	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
5	0	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0
9	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0
43	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
46	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0
14	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0
19	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0
12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	1
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
38	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
32	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0
45	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
42	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

**Table 23** Chandrasekharan and Rajagopalan (1989) [6] 40 parts and 24 machines solution

	21	13	22	1	18	6	8	12	15	4	16	3	20	5	2	11	19	17	10	9	7	14	23	24
17	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1



**Table 25** Chandrasekharan and Rajagopalan (1989) [6] 40 parts and 24 machines solution

	17	9	10	6	12	18	8	15	4	14	23	16	7	24	22	13	21	2	5	19	11	1	3	20
40	<b>1</b>	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
20	<b>1</b>	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
29	<b>1</b>	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
30	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
27	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	1	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	<b>1</b>	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	1	1	0	1	1	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	1	1	1	0	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	<b>1</b>	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1
38	0	0	0	1	0	0	0	0	<b>1</b>	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	<b>1</b>	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	<b>1</b>	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	<b>1</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	<b>1</b>	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	0	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	0	0	1	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	1	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	1	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0	1	0	0
33	0	1	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	1	0	0	0	1	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	1	0	0	0
14	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>	<b>1</b>	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	<b>1</b>	<b>1</b>	<b>1</b>	1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	<b>1</b>	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	<b>1</b>	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	<b>1</b>	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	<b>1</b>	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>
2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>
24	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<b>1</b>	<b>1</b>



## References

1. Ghosh, S., Mahanti, A., Nagi, R., Nau, D.S.: Cell formation problem solved exactly with the Dinkelbach algorithm. *Ann. Oper. Res.* **65**(1), 35–54 (1996)
2. Boe, W., Cheng, C.H.: A close neighbor algorithm for designing cellular manufacturing systems. *Int. J. Prod. Res.* **29**(10), 2097–2116 (1991)
3. Burbidge, J.L.: The new approach to production. *SPE Prod. Eng.* **40**(12), 769–784 (1961)
4. Burbidge, J.L.: *The Introduction of Group Technology*. Wiley, New York (1975)
5. Carrie, S.: Numerical taxonomy applied to group technology and plant layout. *Int. J. Prod. Res.* **11**, 399–416 (1973)
6. Chandrasekharan, M.P., Rajagopalan, R.: Groupability: analysis of the properties of binary data matrices for group technology. *Int. J. Prod. Res.* **27**(6), 1035–1052 (1989)
7. Flanders, R.E.: Design manufacture and production control of a standard machine. *Trans. Am. Soc. Mech. Eng.* **46**, 691–738 (1925)
8. Goldengorin, B., Krushinsky, D., Slomp, J.: Flexible PMP approach for large size cell formation. *Oper. Res.* **60**(5), 1157–1166 (2012)
9. Mitrofanov, S.P.: *Nauchnye Osnovy Gruppovoy Tekhnologii*. Lenizdat, Leningrad (1959), 435 pp. (in Russian)
10. Mitrofanov, S.P.: *The Scientific Principles of Group Technology*. Boston Spa, Yorkshire (1966). National Lending Library Translation, translation of Mitrofanov (1959)
11. Mosier, C.T., Taube, L.: Weighted similarity measure heuristics for the group technology machine clustering problem. *Omega* **13**(6), 577–583 (1985)
12. Seifoddini, H., Wolfe, P.M.: Application of the similarity coefficient method in group technology. *IEE Trans.* **18**(3), 271–277 (1986)
13. Waghodekar, P.H., Sahu, S.: Machine-component cell formation in group technology MACE. *Int. J. Prod. Res.* **22**, 937–948 (1984)
14. Sandbothe, R.A.: Two observations on the grouping efficacy measure for goodness of block diagonal forms. *Int. J. Prod. Res.* **36**, 3217–3222 (1998)
15. Ahi, A., Aryanezhad, Mir.B., Ashtiani, B., Makui, A.: A novel approach to determine cell formation, intracellular machine layout and cell layout in the CMS problem based on TOPSIS method. *Comput. Oper. Res.* **36**(5), 1478–1496 (2009)
16. Bhatnagar, R., Saddikuti, V.: Models for cellular manufacturing systems design: matching processing requirements and operator capabilities. *J. Oper. Res. Soc.* **61**, 827–839 (2010)



# Efficiency Analysis of Branch Network

Petr A. Koldanov

**Abstract** The problem of comparison of several branches efficiency is formulated as a multiple decision problem. The main difficulty to handle this problem lies in the compatibility condition. Solution of this difficulty, based on a method of combination of testing compatible generating hypotheses is given. The additivity condition of the loss function is investigated. This condition is used to construct the optimal multiple decision statistical procedures for comparison of network branches efficiency. Some examples are given.

**Keywords** Branch network · Multiple decision problem · Testing hypothesis · Additivity of the loss function · Unbiased tests

## 1 Introduction

Consider the activity of an organization having branch network. Let these branches operate in different regional centers, under control of one common center and follow common strategy. One of the problems of control consists in comparison of several branches activity efficiency. This information is necessary, in particular, to select a rational method of resources distribution, namely, into which branches resources should be allocated.

Solution of this problem can be based on comparison of numerical indicators of these branches efficiency. In the present paper, such indicator is the quantity of sales (or consumer demand) on corresponding product in different branches. We assume, that the quantities of sales are random variables, distribution of which depends on some unknown parameters. Let the comparison between parameters define the preferences of branches in efficiency. The main problem now is to construct a rule of ranking of branches on the base of set of samples of a small volume.

We consider such problem from mathematical statistic point of view. Our solution is based on the Lehmann's theory of multiple decision statistical procedure

---

P.A. Koldanov (✉)

National Research University Higher School of Economics, Bolshaya Pecherskaya, 25/12,  
Nizhny Novgorod, 603155, Russian Federation  
e-mail: [pkoldanov@hse.ru](mailto:pkoldanov@hse.ru)

**Table 1** The numbers of sales and total number of customers

Branch	1	2	3	4	5	6	7	8
2000	254	199	187	182				
2001	238	381	305	148	193	99	235	
2002	386	447	612	205	233	396	444	101
2003	328	445	700	247	223	563	400	129
2004	393	388	706	261	223	570	465	119
2005	447	376	685	284	235	479	364	102
2006	568	390	726	374	205	404	348	111
2007	432	302	651	391	260	532	438	54
NUM	6390	7090	28900	6320	6320	11130	4660	2530

[1] and tests of the Neyman structure [2]. Lehmann's theory of multiple decision statistical procedures is based on three points: choice of the generating hypothesis, compatibility of the family of tests for the generating hypothesis with decision space for the original hypothesis, and additivity of the loss function.

In the present paper, we formulate the problem of comparison of branch activity efficiency as a multiple decision problem. Then we discuss the choice of generating hypothesis and compatibility condition, and investigate the condition of additivity of the loss function and condition of unbiasedness.

The paper is organized as follow. In Sect. 2, we give an example of real trading (distributive) network that we will use in the paper to illustrate the theoretical findings. In Sect. 3, mathematical formulation of the problem is presented. In Sect. 4, multiple decision procedure for the solution of the stated problem is described. In Sect. 5, the condition of additivity of the loss function and condition of unbiasedness are analyzed. In the last Sect. 6, we summarize the main results of the paper.

## 2 Example of Real Distribution Network

Let's consider activity of a trading network with the main office in the center of region and with branches in different regional cities. We suppose this trading network merchandises same expensive product, for example, cars. In this case, the ratio between number of sales and total number of customers will be an efficiency indicator. Let's assume that such network has worked for some years (as a whole structure) and the management team of the network has a problem of choice of rational strategy of the network development. Comparison of results of branches efficiency is necessary to choose such strategy.

The numbers of sales and total number of customers are presented in Table 1. Some of branches started to work later, namely the branches 5, 6, 7 started to work in 2001 and the branch 8 started to work in 2002. String NUM corresponds to the total number of customers.

Suppose that the numbers of sales are random variables  $X_i$  ( $i = 1, \dots, 8$ ). In this case, the data in the table  $x_j^i$  ( $i = 1, \dots, N$ ;  $j = 1, \dots, m_j$ ) are observations of

these random variables. As indicator of branch efficiency we will consider the ratio between number of those who make a purchase to potentially possible number of buyers in corresponding city.

### 3 Mathematical Model and Formulation of the Problem

Let  $N$  be the number of branches,  $n_j^i$  be the potential number of buyers in the city  $i$  for the year  $j$ ,  $m_i$  be the number of observations for the city  $i$ ,  $i = 1, 2, \dots, N$ . Define the random variables with Bernoulli distributions

$$\xi_{kj}^i = \begin{cases} 0, & P(\xi_{kj}^i = 0) = q_{kj}^i \\ 1, & P(\xi_{kj}^i = 1) = p_{kj}^i \end{cases} \quad (1)$$

$p_{kj}^i + q_{kj}^i = 1$ ;  $k = 1, \dots, n^i$ ;  $i = 1, \dots, N$ ;  $j = 1, \dots, m_i$ . In our setting, the random variable  $\xi_{kj}^i$  describes behavior of buyer number  $k$  of the city  $i$  for the year  $j$ ,  $p_{kj}^i$  is the probability of the fact that the buyer  $k$  from the city  $i$  make a purchase in year  $j$ . Therefore, the random variable

$$X_j^i = \sum_{k=1}^{n^i} \xi_{kj}^i$$

describes number of sales in the city  $i$  for the year  $j$ . We use the following notations  $a_j^i = E(X_j^i) = \sum_{k=1}^{n^i} p_{kj}^i$ ,  $X^i = (X_1^i, X_2^i, \dots, X_{m_i}^i)$ ,  $x^i = (x_1^i, x_2^i, \dots, x_{m_i}^i)$ . The problem is investigated in the paper under the following assumptions:

- $n_j^i = n^i$ ,  $\forall j = 1, \dots, m_i$ , where  $m_i$ —number of years of observations for the branch  $i$ . Let  $n^i$  be known (it can be a fixed percent of the city population).
- $a_j^i = a^i$  ( $j = 1, \dots, m_i$ ).
- The random variables  $X_j^i$  are independent for  $j = 1, \dots, m_i$ ;  $i = 1, \dots, N$ .
- The random variable  $X_j^i$  has a normal distribution from the class  $N(a^i, \sigma^{i^2}) \forall j = 1, \dots, m_i$ .
- Indicator of efficiency (consumer demand) for the city  $i$  is calculated by:

$$p^i = \frac{a^i}{n^i} = \sum_{k=1}^{n^i} \frac{p_k^i}{n^i}$$

- Relation between parameters is:

$$\frac{\sigma^{i^2}}{\sigma^{j^2}} = \frac{n^i}{n^j}$$

Discussion of this assumption is given in [3].

The problem of ranking of the branches efficiency can be formulated as multiple decision problem of choice from  $L$  hypothesis:

$$\begin{aligned}
 H_1: p^1 &= p^2 = \dots = p^N \\
 H_2: p^1 &> p^2 = \dots = p^N \\
 H_3: p^1 &< p^2 = \dots = p^N \\
 H_4: p^1 &= p^2 > p^3 = \dots = p^N \\
 &\vdots \\
 H_L: p^1 &< p^2 < \dots < p^N
 \end{aligned} \tag{2}$$

Hypothesis  $H_1$  means that efficiencies for all branches are equal. Hypothesis  $H_2$  means that the branch 1 is the most efficient and all other branches have equal efficiencies and so on. Hypothesis  $H_L$  means that the branches are ranked in efficiency by their ordinal number.

Note, that the total number of hypothesis is:

$$L = \sum_{r=1}^N \sum_{k_1+k_2+\dots+k_r=N} \frac{N!}{k_1!k_2!\dots k_r!} \tag{3}$$

This number is increasing very fast. If  $N = 2$  then  $L = 3$ , if  $N = 3$  then  $L = 13$ , if  $N = 4$  then  $L = 75$ , if  $N = 5$  then  $L = 541$  and so on.

## 4 Compatibility Conditions and Multiple Decision Statistical Procedure

The problem of choice of one of  $L$  hypotheses can be studied in the framework of the theory of statistical decision functions [4, 5]. In [1], a constructive way for the solution is given. Research in this and similar directions is proceeded till now. The detailed bibliography is presented in [6, 7]. An application of the multiple decision theory to the market graph construction is given in [8].

Main objective of the present paper is application of the method from [1] for the solution of the problem of branch ranking and illustration of adequacy of this method to this type of problems.

To construct the test with  $L$  decisions for the problem (2) we can apply the method described in [1], where the test with  $L$  decisions is constructed from the tests with  $K < L$  decisions. One can use a natural tests with  $K = 3$ , therefore the problem with  $L$  decisions is reduced to set of problems with three decisions. The last problem can be reduced to two usual testing hypothesis problems [2].

Consider the following sets of three decisions problems (three-decisions generating problems for (2)):

$$H_1^{ij}: p^i < p^j, \quad H_2^{ij}: p^i = p^j, \quad H_3^{ij}: p^i > p^j \tag{4}$$

where  $p^i, p^j$  are indicators of efficiency for the pair of branches  $(i, j)$ . According to [1], the following test for the problem (4) can be applied (see [3] for more details):

$$\delta_{ij}(x^i, x^j) = \begin{cases} d_1^{ij}, & \text{if } t_{ij}(x^i, x^j) < c_1 \\ d_2^{ij}, & \text{if } c_1 \leq t_{ij}(x^i, x^j) \leq c_2 \\ d_3^{ij}, & \text{if } t_{ij}(x^i, x^j) > c_2 \end{cases} \quad (5)$$

where the statistic of test  $t_{ij}$  has the form:

$$t_{ij}(x^i, x^j) = \frac{(\bar{x}^i/n^i - \bar{x}^j/n^j) / \sqrt{\frac{1}{m_i n^i} + \frac{1}{m_j n^j}}}{\sqrt{(\sum_{i=1}^N \frac{1}{n^i} \sum_{l=1}^{m_i} (x_l^i - \bar{x}^i)^2) / (\sum_{i=1}^N m_i - N)}} \quad (6)$$

Here  $d_k^{ij}$  is the decision of acceptance of  $H_k^{ij}$ ,  $k = 1, 2, 3$ ;  $\bar{x}^i = \frac{1}{m_i} \sum_{j=1}^{m_i} x_j^i$ . The constants  $c_1, c_2$  are defined from the equations:

$$\begin{aligned} P(t_{ij}(X^i, X^j) < c_1 | p^i = p^j) &= \alpha_{ij} \\ P(t_{ij}(X^i, X^j) > c_2 | p^i = p^j) &= \alpha_{ji} \end{aligned} \quad (7)$$

The problem (4) is a three-decision problem. To construct the test (5), we use the following generating hypotheses (see details in [3]):

$$\begin{aligned} h_{ij}: p^i \geq p^j \quad \text{vs.} \quad k_{ij}: p^i < p^j \\ h_{ji}: p^j \geq p^i \quad \text{vs.} \quad k_{ji}: p^j < p^i \end{aligned} \quad (8)$$

and combine two well known unbiased tests for them.

The test (5) is valid under the additional condition of compatibility of the tests for generating hypotheses which reads in our case as

$$P_\theta(\delta_{ij}(x) = d_1) + P_\theta(\delta_{ij}(x) = d_2) + P_\theta(\delta_{ij}(x) = d_3) = 1, \quad \theta = p^i - p^j$$

This is equivalent to  $c_1 < c_2$  i.e.  $\alpha_{ij} + \alpha_{ji} < 1$  (see [1], [8] where the compatibility condition is discussed). For the case  $\alpha_{ij} = \alpha_{ji}$  we have  $c_2 = -c_1$ . Now we consider the compatibility conditions for the problem (2) with generating hypotheses (4). According to Wald decision theory [4], a nonrandomized multiple decision statistical procedure for the problem (2) is a partition of the sample space into  $L$  regions. The construction of the test for the  $L$ -decision problem (2) from the tests (5) faces the problem of compatibility which does not have a solution in our case. Indeed consider the problem (2) for the case  $N = 3$  and  $\alpha_{ij} = \alpha$ ,  $\forall i, j$ . Then we have 13 regions in parametric space (13 hypothesis):

$$\begin{aligned} H_1: p^1 = p^2 = p^3, & \quad H_2: p^1 > p^2 = p^3, & \quad H_3: p^1 < p^2 = p^3 \\ H_4: p^1 = p^2 > p^3, & \quad H_5: p^1 = p^2 < p^3, & \quad H_6: p^1 = p^3 < p^2 \\ H_7: p^1 = p^3 > p^2, & \quad H_8: p^1 < p^2 < p^3, & \quad H_9: p^1 < p^3 < p^2 \\ H_{10}: p^2 < p^1 < p^3, & \quad H_{11}: p^2 < p^3 < p^1, & \quad H_{12}: p^3 < p^1 < p^2 \\ H_{13}: p^3 < p^2 < p^1 \end{aligned} \quad (9)$$

If we combine the tests (5), then the sample space is divided onto  $3^3 = 27$  regions, which are (we put  $c = c_2 = -c_1$ ):

$$\begin{aligned}
 D_1 &= \begin{Bmatrix} t_{12} < -c \\ t_{13} < -c \\ t_{23} < -c \end{Bmatrix}; & D_2 &= \begin{Bmatrix} t_{12} < -c \\ t_{13} < -c \\ |t_{23}| \leq c \end{Bmatrix}; & D_3 &= \begin{Bmatrix} t_{12} < -c \\ t_{13} < -c \\ t_{23} > c \end{Bmatrix} \\
 D_4 &= \begin{Bmatrix} t_{12} < -c \\ |t_{13}| \leq c \\ t_{23} < -c \end{Bmatrix}; & D_5 &= \begin{Bmatrix} t_{12} < -c \\ |t_{13}| \leq c \\ |t_{23}| \leq c \end{Bmatrix}; & D_6 &= \begin{Bmatrix} t_{12} < -c \\ |t_{13}| \leq c \\ t_{23} > c \end{Bmatrix} \\
 D_7 &= \begin{Bmatrix} t_{12} < -c \\ t_{13} > c \\ t_{23} < -c \end{Bmatrix}; & D_8 &= \begin{Bmatrix} t_{12} < -c \\ t_{13} > c \\ |t_{23}| \leq c \end{Bmatrix}; & D_9 &= \begin{Bmatrix} t_{12} < -c \\ t_{13} > c \\ t_{23} > c \end{Bmatrix} \\
 D_{10} &= \begin{Bmatrix} |t_{12}| \leq c \\ t_{13} < -c \\ t_{23} < -c \end{Bmatrix}; & D_{11} &= \begin{Bmatrix} |t_{12}| \leq c \\ t_{13} < -c \\ |t_{23}| \leq c \end{Bmatrix}; & D_{12} &= \begin{Bmatrix} |t_{12}| \leq c \\ t_{13} < -c \\ t_{23} > c \end{Bmatrix} \\
 D_{13} &= \begin{Bmatrix} |t_{12}| \leq c \\ |t_{13}| \leq c \\ t_{23} < -c \end{Bmatrix}; & D_{14} &= \begin{Bmatrix} |t_{12}| \leq c \\ |t_{13}| \leq c \\ |t_{23}| \leq c \end{Bmatrix}; & D_{15} &= \begin{Bmatrix} |t_{12}| \leq c \\ |t_{13}| \leq c \\ t_{23} > c \end{Bmatrix} \\
 D_{16} &= \begin{Bmatrix} |t_{12}| \leq c \\ t_{13} > c \\ t_{23} < -c \end{Bmatrix}; & D_{17} &= \begin{Bmatrix} |t_{12}| \leq c \\ t_{13} > c \\ |t_{23}| \leq c \end{Bmatrix}; & D_{18} &= \begin{Bmatrix} |t_{12}| \leq c \\ t_{13} > c \\ t_{23} > c \end{Bmatrix} \\
 D_{19} &= \begin{Bmatrix} t_{12} > c \\ t_{13} < -c \\ t_{23} < -c \end{Bmatrix}; & D_{20} &= \begin{Bmatrix} t_{12} > c \\ t_{13} < -c \\ |t_{23}| \leq c \end{Bmatrix}; & D_{21} &= \begin{Bmatrix} t_{12} > c \\ t_{13} < -c \\ t_{23} > c \end{Bmatrix} \\
 D_{22} &= \begin{Bmatrix} t_{12} > c \\ |t_{13}| \leq c \\ t_{23} < -c \end{Bmatrix}; & D_{23} &= \begin{Bmatrix} t_{12} > c \\ |t_{13}| \leq c \\ |t_{23}| \leq c \end{Bmatrix}; & D_{24} &= \begin{Bmatrix} t_{12} > c \\ |t_{13}| \leq c \\ t_{23} > c \end{Bmatrix} \\
 D_{25} &= \begin{Bmatrix} t_{12} > c \\ t_{13} > c \\ t_{23} < -c \end{Bmatrix}; & D_{26} &= \begin{Bmatrix} t_{12} > c \\ t_{13} > c \\ |t_{23}| \leq c \end{Bmatrix}; & D_{27} &= \begin{Bmatrix} t_{12} > c \\ t_{13} > c \\ t_{23} > c \end{Bmatrix}
 \end{aligned}$$

One has from (6) under additional assumption that  $m_i = m$ ,  $n^i = n$ ,  $i = 1, 2, \dots, N$ :

$$\begin{aligned}
 t_{13} &= t_{12} + t_{23} \\
 t_{12} &= t_{13} + t_{32} \\
 t_{23} &= t_{21} + t_{13}
 \end{aligned} \tag{10}$$

Therefore, the sample regions  $D_4, D_7, D_8, D_{12}, D_{16}, D_{20}, D_{21}, D_{24}$  are empty. This means that decision function induced by (5) consists of  $27 - 8 = 19$  decisions. On the other side, decision function for the problem (9) has to have 13 decisions only. Note, that the regions  $D_5, D_{11}, D_{13}, D_{15}, D_{17}, D_{23}$  in the sample space does not have a corresponding nonempty regions in the parametric space. For example, the

region  $D_5$  in the sample space corresponds to the empty region  $p^1 < p^2$ ;  $p^1 = p^3$ ;  $p^2 = p^3$  in the parametric space. At the same time, one has (it is true in general case too)

$$P(|t_{ij}| \leq c, |t_{jk}| \leq c, |t_{ik}| > c) > 0$$

which means that the probability to accept the decision  $p^i = p^j$ ;  $p^j = p^k$  but  $p^i \neq p^k$  is not equal to zero.

To handle the problem of incompatibility, we reformulate the initial ranking problem (2) in the following way. First, we introduce the notations

$$\begin{aligned} p^i &\stackrel{\Delta}{=} p^j &\iff & |p^i - p^j| \leq \Delta \\ p^i &\stackrel{\Delta}{<} p^j &\iff & p^i + \Delta < p^j \\ p^i &\stackrel{\Delta}{>} p^j &\iff & p^i > p^j + \Delta \end{aligned}$$

where  $\Delta$  is fixed positive number. Next, we formulate new multiple decision problem of the choice from  $M$  hypotheses:

$$\begin{aligned} H'_1: p^i &\stackrel{\Delta}{=} p^j, \quad \forall i, j = 1, \dots, N \\ H'_2: p^1 &\stackrel{\Delta}{>} p^i, \quad i = 2, \dots, N; \quad p^i &\stackrel{\Delta}{=} p^j, \quad \forall i, j = 2, \dots, N \\ H'_3: p^1 &\stackrel{\Delta}{>} p^2; p^2 &\stackrel{\Delta}{>} p^i, \quad i = 3, \dots, N; \quad p^i &\stackrel{\Delta}{=} p^j, \quad \forall i, j = 3, \dots, N \\ &\vdots \\ H'_M: p^1 &\stackrel{\Delta}{<} p^2; \quad p^2 &\stackrel{\Delta}{<} p^3; \quad \dots, \quad p^{N-1} &\stackrel{\Delta}{<} p^N \end{aligned} \quad (11)$$

To solve the problem (11), we introduce the following set of three-decisions generating problems

$$\begin{aligned} H_1^{(ij)}: p^i &\stackrel{\Delta}{<} p^j \\ H_2^{(ij)}: p^i &\stackrel{\Delta}{=} p^j \\ H_3^{(ij)}: p^i &\stackrel{\Delta}{>} p^j \end{aligned} \quad (12)$$

The test (5) can be applied for the problem (12) with the constants  $c_1, c_2$  defined from the equations:

$$\begin{aligned} P(T_{ij}(X^i, X^j) < c_1 | p^i + \Delta = p^j) &= \alpha_{ij} \\ P(T_{ij}(X^i, X^j) > c_2 | p^i = p^j + \Delta) &= \alpha_{ji} \end{aligned} \quad (13)$$

**Table 2** The results of testing the three-decisions generating problems (12)

	1	2	3	4	5	6
2	$ p^1 - p^2  < \Delta$					
3	$p^1 > p^3 + \Delta$	$p^2 > p^3 + \Delta$				
4	$p^1 > p^4 + \Delta$	$ p^2 - p^4  < \Delta$	$p^4 > p^3 + \Delta$			
5	$p^1 > p^5 + \Delta$	$p^2 > p^5 + \Delta$	$ p^3 - p^5  < \Delta$	$ p^4 - p^5  < \Delta$		
6	$p^1 > p^6 + \Delta$	$p^2 > p^6 + \Delta$	$p^6 > p^3 + \Delta$	$ p^4 - p^6  < \Delta$	$ p^5 - p^6  < \Delta$	
7	$p^7 > p^1 + \Delta$	$p^7 > p^2 + \Delta$	$p^7 > p^3 + \Delta$	$p^7 > p^4 + \Delta$	$p^7 > p^5 + \Delta$	$p^7 > p^6 + \Delta$
8	$ p^1 - p^8  < \Delta$	$ p^2 - p^8  < \Delta$	$p^8 > p^3 + \Delta$	$ p^4 - p^8  < \Delta$	$ p^5 - p^8  < \Delta$	$ p^6 - p^8  < \Delta$
7-8						$p^7 > p^8 + \Delta$

In this setting, there is a one to one correspondence between associated partition regions in the parameters and sample spaces. For example for the case  $N = 3$ , the problem (11) can be written as:

$$\begin{aligned}
H'_1: p^1 \trianglelefteq p^2 \trianglelefteq p^3 & \quad H'_2: p^1 \trianglelefteq p^2 \trianglelefteq p^3 & H'_3: p^1 \trianglelefteq p^3 \trianglelefteq p^2 \\
H'_5: p^1 \trianglelefteq p^3 \trianglelefteq p^2 & \quad H'_6: p^1 \trianglelefteq p^3 \trianglelefteq p^2 & H'_9: p^3 \trianglelefteq p^1 \trianglelefteq p^2 \\
H'_{10}: p^1 \trianglelefteq p^2 \trianglelefteq p^3 & \quad H'_{11}: p^1 \trianglelefteq p^2 \trianglelefteq p^3 & H'_{13}: p^2 \trianglelefteq p^1 \trianglelefteq p^3 \\
H'_{14}: p^1 \trianglelefteq p^3 \trianglelefteq p^2 & \quad H'_{15}: p^3 \trianglelefteq p^1 \trianglelefteq p^2 & H'_{17}: p^3 \trianglelefteq p^2 \trianglelefteq p^1 \\
H'_{18}: p^3 \trianglelefteq p^1 \trianglelefteq p^2 & \quad H'_{19}: p^2 \trianglelefteq p^1 \trianglelefteq p^3 & H'_{22}: p^2 \trianglelefteq p^1 \trianglelefteq p^3 \\
H'_{23}: p^2 \trianglelefteq p^3 \trianglelefteq p^1 & \quad H'_{25}: p^2 \trianglelefteq p^3 \trianglelefteq p^1 & H'_{26}: p^2 \trianglelefteq p^3 \trianglelefteq p^1 \\
H'_{27}: p^3 \trianglelefteq p^2 \trianglelefteq p^1 & & 
\end{aligned} \tag{14}$$

where  $p^i \trianglelefteq p^j \trianglelefteq p^k$  means  $|p^i - p^k| < \Delta$ ,  $|p^j - p^k| < \Delta$  and  $p^i + \Delta < p^j$ . It is easy to see that there exists one-to-one correspondence  $D_i \longleftrightarrow H'_i$ , between partition of the sample space (see above) and partition of the parametric space (14).

Note that number  $M$  of hypothesis in (11) is larger than the number of hypothesis in (2).

We illustrate our findings by the practical example.

*Example* Results of application of the multiple decision problem (11) for the data from Table 1,  $\Delta = 10^{-6}$  are given in Table 2.



According to (11), accepted decisions are:

$$\begin{aligned}
 p^1 &> p^i + \Delta, \quad i = 3, 4, 5, 6; \\
 p^2 &> p^i + \Delta, \quad i = 3, 5, 6; \\
 p^4 &> p^3 + \Delta; \\
 p^6 &> p^3 + \Delta; \\
 p^7 &> p^i + \Delta, \quad i = 1, 2, 3, 4, 5, 6, 8; \\
 |p^i - p^j| &< \Delta, \quad (i, j) = (1, 2), (1, 8), (2, 4), (2, 8), (3, 5), (4, 5), \\
 &\quad (4, 6), (4, 8), (5, 6), (5, 8), (6, 8)
 \end{aligned} \tag{15}$$

The general conclusion about indicators of branches efficiency can be written as follows:

$$p^3 \stackrel{\Delta}{\leq} p^5 \stackrel{\Delta}{\leq} p^6 \stackrel{\Delta}{\leq} p^4 \stackrel{\Delta}{\leq} p^8 \stackrel{\Delta}{\leq} p^2 \stackrel{\Delta}{\leq} p^1 \stackrel{\Delta}{\leq} p^7 \tag{16}$$

## 5 Statistical Optimality of Ranking

In this section, we discuss some properties of the constructed multiple decision statistical procedure. In particular, we show that this procedure is optimal in the class of unbiased multiple decision statistical procedures. To prove this fact, we follow the method proposed in [1].

First, we show that the test (5) with constant defined by (13) is optimal in the class of unbiased statistical procedure for the problem (12).

Generating hypothesis for the problem (12) are:

$$\begin{aligned}
 h'_{ij}: p^i \leq p^j + \Delta \quad \text{vs.} \quad k'_{ij}: p^i > p^j + \Delta \\
 h'_{ji}: p^j \leq p^i + \Delta \quad \text{vs.} \quad k'_{ji}: p^j > p^i + \Delta
 \end{aligned} \tag{17}$$

Uniformly most powerful unbiased tests for problems (17) are [2]:

$$\delta'_{ij}(x^i, x^j) = \begin{cases} d'_{ij}, & \text{if } t_{ij}(x^i, x^j) > c_2 \\ d_{ij}, & \text{if } t_{ij}(x^i, x^j) < c_2 \end{cases} \tag{18}$$

$$\delta'_{ji}(x^i, x^j) = \begin{cases} d'_{ji}, & \text{if } t_{ji}(x^i, x^j) > c_1 \\ d_{ji}, & \text{if } t_{ji}(x^i, x^j) < c_1 \end{cases} \tag{19}$$

where  $d'_{ij}(d_{ij})$ —decision of rejection (acceptance) of hypothesis  $h'_{ij}$  and constant  $c_1, c_2$  are defined by (13). If  $\alpha_{ij} + \alpha_{ji} \leq 1$ , then the compatibility condition is satisfied (see Sect. 4). Note that power functions of tests (18)–(19) are continuous.

Consider the loss function for the three decision test (5). To simplify our arguments, we drop the index  $(i, j)$  in the notations. Let  $w_{lk}$  be the loss from the

acceptance of decision  $d_k$  when  $H'_l$  is true,  $l, k = 1, 2, 3$ ,  $w_{ll} = 0$ ,  $a_1, a_2$  be the loss from the rejection of  $h'_{ij}$  and  $h'_{ji}$  when they are true,  $b_1, b_2$  be the loss from the acceptance of  $h'_{ij}$  and  $h'_{ji}$  when  $k'_{ij}$  and  $k'_{ji}$  are true. We can evaluate the losses as follows. Suppose the company has a fund  $s$  to be invested in the development of branches. The investment strategy is to invest in the most efficient branch and divide the investment if they are equal in efficiency. In this case, if the hypothesis  $H'_1$  is true then the losses from decisions  $d_1, d_2, d_3$  are  $w_{11} = 0$ ,  $w_{1,2} = s/2$ ,  $w_{13} = s$ . If the hypothesis  $H'_2$  is true, then the losses from decisions  $d_1, d_2, d_3$  are  $w_{21} = s/2$ ,  $w_{2,2} = 0$ ,  $w_{23} = s/2$ . If the hypothesis  $H'_3$  is true, then the losses from decisions  $d_1, d_2, d_3$  are  $w_{31} = s$ ,  $w_{3,2} = s/2$ ,  $w_{33} = 0$ . Therefore, the following relations take place:

$$w_{12} = b_2; \quad w_{13} = a_1 + b_2; \quad w_{23} = a_1; \quad w_{21} = a_2; \quad w_{31} = a_2 + b_1; \quad w_{32} = b_1 \quad (20)$$

This is exactly the additivity conditions for the loss function in [1]. The additivity conditions imply

$$w_{13} = w_{12} + w_{23}; \quad w_{31} = w_{32} + w_{21} \quad (21)$$

Now we state that compatibility condition for generating hypotheses testing is satisfied and additivity of the loss function takes place and power functions of tests (18)–(19) are continuous. Therefore, combining most powerful unbiased tests with two decisions we get a optimal statistical procedure in the class of unbiased statistical procedures with three decisions.

Next step is to consider the general problem (11)–(12). It was shown that tests (5) for the problem (11) are compatible and are optimal statistical procedure in the class of unbiased statistical procedures with three decisions. Condition of additivity of the loss function is:

$$w(\Theta, \delta) = \sum_{i < j} w(\Theta, \delta_{ij}) \quad (22)$$

where  $\delta_{ij}$  is the statistical procedure (5) for the problem (12) with constants defined by (13),  $\Theta = (p^1, \dots, p^N)$ . This condition means that the total loss is a sum of losses from statistical procedures for generating three decision problems.

To illustrate the relations (22), we consider the case  $N = 3$ ; the general case can be treated in the same way. The multiple decision problem (11) for the case  $N = 3$  is (14) where  $M = 19$ .

**Case 1.** True decision and taken decision are different in one pair of branches. One adjacent hypotheses error. For example, suppose the hypothesis  $H'_{14}$  is true, but the decision  $d'_{11}$  is accepted. In this case one has  $w_{14,11} = w_{21}^{13}$ . This is the loss from the acceptance of wrong decision  $p^1 \triangleleft p^3$  when  $p^1 \triangleleft p^3$  is true. The intersection of closures of parametric domains for this two hypotheses is not empty. We call this type of error as adjacent hypotheses error. This type of error can be coded by 1–2 (hypotheses  $H_1$  and  $H_2$ ), 2–1 (hypotheses  $H_2$  and  $H_1$ ), 2–3 (hypotheses  $H_2$  and  $H_3$ ), 3–2 (hypotheses  $H_3$  and  $H_2$ ).

- Case 2.** True decision and taken decision are different in one pair of branches. One separated hypotheses error. For example, suppose the hypothesis  $H'_1$  is true, but the decision  $d'_3$  is accepted. In this case, one has  $w_{1,3} = w_{13}^{23}$ . This is the loss from the acceptance of wrong decision  $p^3 \triangleleft p^2$  when  $p^2 \triangleleft p^3$  is true. The intersection of closures of parametric domains for this two hypotheses is empty. We call this type of error as separated hypotheses error. This type of error can be coded by 3–1 (hypotheses  $H_3$  and  $H_1$ ), 1–3 (hypotheses  $H_1$  and  $H_3$ ). Note that this type of error is more serious than adjacent hypotheses error.
- Case 3.** True decision and taken decision are different in two pairs of branches. Two adjacent hypotheses errors. For example, suppose the hypothesis  $H'_1$  is true, but the decision  $d'_{11}$  is taken. In this case, one has from additivity condition  $w_{1,11} = w_{12}^{12} + w_{12}^{23}$ . This means that we have the errors of the type 1–2 in the comparison of branches 1 and 2 and the error of the type 1–2 in the comparison of branches 2 and 3.
- Case 4.** True decision and taken decision are different in two pairs of branches. Two separated hypotheses errors. For example, suppose the hypothesis  $H'_2$  is true, but the decision  $d'_{26}$  is taken. In this case, one has  $w_{2,26} = w_{13}^{13} + w_{13}^{21}$ . This means that we have the errors of the type 1–3 in the comparison of branches 1 and 3 and the error of the type 1–3 in the comparison of branches 2 and 1.
- Case 5.** True decision and taken decision are different in two pairs of branches. One adjacent hypotheses and one separated hypotheses errors. For example, suppose the hypothesis  $H'_2$  is true, but the decision  $d'_{23}$  is taken. In this case, one has  $w_{2,23} = w_{13}^{12} + w_{12}^{13}$ . This means that we have the errors of the type 1–3 in the comparison of branches 1 and 2 and the error of the type 1–2 in the comparison of branches 1 and 3.
- Case 6.** True decision and taken decision are different in three pairs of branches. Two adjacent hypotheses and one separated hypotheses errors. For example, suppose the hypothesis  $H'_1$  is true, but the decision  $d'_{23}$  is taken. In this case one has  $w_{1,23} = w_{13}^{12} + w_{12}^{13} + w_{12}^{23}$ . This means that we have the errors of the type 1–3 in the comparison of branches 1 and 2, the error of the type 1–2 in the comparison of branches 1 and 3 and the error of the type 1–2 in the comparison of branches 2 and 3.
- Case 7.** True decision and taken decision are different in three pairs of branches. Three separated hypotheses errors. For example, suppose the hypothesis  $H'_1$  is true, but the decision  $d'_{27}$  is taken. In this case, one has  $w_{1,27} = w_{13}^{12} + w_{13}^{13} + w_{13}^{23}$ . This means that we have the errors of the type 1–3 in the comparison of branches 1 and 2, the error of the type 1–3 in the comparison of branches 1 and 3 and the error of the type 1–3 in the comparison of branches 2 and 3.

Condition of additivity of the loss function in our problem means that the larger weight is attached to the losses resulting from taken decision being far from the true decision. Now we state that compatibility condition for generating hypotheses testing for the problem (11) is satisfied and additivity of the loss function takes place. Therefore, combining optimal unbiased statistical procedures with three decisions

we get a optimal statistical procedure in the class of unbiased statistical procedures with  $M$  decisions.

We end this section by some discussion on the unbiasedness in three decision case. Following [2], we call the statistical procedure  $\delta(x)$  unbiased, if for any  $\theta, \theta' \in \Omega$

$$E_{\theta} w(\theta', \delta(x)) \geq E_{\theta} w(\theta, \delta(x)) \quad (23)$$

In our case  $\theta = p^i - p^j$ . For the statistical procedure (5), conditional risk is:

$$E_{\theta}(\theta, \delta(x)) = \begin{cases} w_{12} P_{\theta}(\delta(x) = d_2) + w_{13} P_{\theta}(\delta(x) = d_3), & \text{if } p^i \triangleleft p^j \\ w_{21} P_{\theta}(\delta(x) = d_1) + w_{23} P_{\theta}(\delta(x) = d_3), & \text{if } p^i \triangleq p^j \\ w_{31} P_{\theta}(\delta(x) = d_1) + w_{32} P_{\theta}(\delta(x) = d_2), & \text{if } p^i \triangleright p^j \end{cases} \quad (24)$$

Therefore, from (21) and the relation

$$P_{\theta}(\delta(x) = d_1) + P_{\theta}(\delta(x) = d_2) + P_{\theta}(\delta(x) = d_3) = 1$$

the statistical procedure  $\delta(x)$  for the problem (12) is unbiased if and only if:

$$\left\{ \begin{array}{ll} P_{\theta}(\delta(x) = d_1) \geq \frac{w_{12}}{w_{12} + w_{21}} = \alpha_{i,j}, & \text{if } p^i \triangleleft p^j \\ P_{\theta}(\delta(x) = d_3) \geq \frac{w_{32}}{w_{23} + w_{32}} = \alpha_{j,i}, & \text{if } p^i \triangleright p^j \\ P_{\theta}(\delta(x) = d_1) \leq \frac{w_{12}}{w_{12} + w_{21}} = \alpha_{i,j}, & \text{if } p^i \triangleq p^j \\ P_{\theta}(\delta(x) = d_3) \leq \frac{w_{32}}{w_{23} + w_{32}} = \alpha_{j,i}, & \text{if } p^i \triangleq p^j \\ P_{\theta}(\delta(x) = d_2) + \frac{w_{31} + w_{13}}{w_{21} + w_{12}} P_{\theta}(\delta(x) = d_3) \leq \frac{w_{31}}{w_{12} + w_{21}} & \text{if } p^i \triangleleft p^j \\ P_{\theta}(\delta(x) = d_2) + \frac{w_{31} + w_{13}}{w_{23} + w_{32}} P_{\theta}(\delta(x) = d_1) \leq \frac{w_{13}}{w_{32} + w_{23}} & \text{if } p^i \triangleright p^j \end{array} \right. \quad (25)$$

First four conditions (25) are usual restrictions on probability of wrong decisions. Last two conditions (25) are restrictions for linear combinations of probabilities of wrong decisions if hypothesis  $H_1^{(ij)}$  or  $H_3^{(ij)}$  are true.

## 6 Conclusions

Problem of ranking of the branch efficiency is considered as a multiple decision problem. Solution of this problem is given on the base of multiple decision theory. As a result, a corresponding multiple decision statistical procedure is constructed. This multiple decision statistical procedure is taken as combination of three-decision statistical procedures. It is shown that this three-decision statistical procedures are optimal in the class of unbiased statistical procedures and as consequence the multiple decision statistical procedure is optimal in the class of unbiased multiple decision statistical procedures.

**Acknowledgements** The author is partly supported by National Research University Higher School of Economics, Russian Federation Government grant, N. 11.G34.31.0057.

## References

1. Lehmann, E.L.: A theory of some multiple decision procedures 1. *Ann. Math. Stat.* **28**, 1–25 (1957)
2. Lehmann, E.L., Romano, J.P.: *Testing Statistical Hypotheses*. Springer, New York (2005)
3. Koldanov, A.P., Koldanov, P.A.: Multiple decision procedures for the analysis of higher school entry selection results. *Bus. Inform.* **1**(19), 24–31 (2012) (in Russian)
4. Wald, A.: *Statistical Decision Function*. Wiley, New York (1950)
5. Anderson, T.W.: *An Introduction to Multivariate Statistical Analysis*. Wiley, New York (1957)
6. Rao, C.V., Swarupchand, U.: Multiple comparison procedures—a note and a bibliography. *J. Stat.* **16**, 66–109 (2009)
7. Shaffer, J.P.: Recent development towards optimality in multiple hypothesis testing. In: *Optimality: The Second Erich L. Lehmann Symposium*. IMS Lecture Notes—Monograph Series, vol. 49, pp. 16–32 (2006)
8. Koldanov, A.P., Koldanov, P.A., Kalyagin, V.A., Pardalos, P.M.: Statistical procedures for the market graph construction. *Comput. Stat. Data Anal.* **68**, 17–29 (2013)

# EEG Coherence in Right- and Left-Handers in Passive Visual Perception of Lines with Different Slope Angles

Maxim Viktorovich Lukoyanov

**Abstract** The level of EEG coherence in right- and left-handers during passive viewing of white screen and lines with different slope angles has been studied. The right-handers had mostly local intra-hemispheric changes of EEG coherence, while these changes in the left-handers were inter-hemispheric.

**Keywords** EEG · Coherence · Left-handers · Inter-hemispheric changes · Intra-hemispheric changes

## 1 Introduction

Comparison of brain activities of left- and right-handers can help us to understand some total principles of brain activity, because preference of one hand but not another reflects different organizations of neural process. EEG coherence which reflects connections between brain areas is informative and well known method for investigation of this problem. EEG coherence is quantitative index of synchronicity involving various cortical areas during their interaction. High coherence means that in two points, where electrical potentials are recorded, brain activities accord on frequency and constant in phase [1]. However, the data obtained by various researches who investigated the coherence of the left-handed and right-handers, give different results. The aim of this work is comparison of the EEG coherence and its changes during passive viewing of a white screen and lines with various slopes by subjects with different hand preference.

---

M.V. Lukoyanov (✉)

Lobachevsky State University, Nizhny Novgorod, Russian Federation

e-mail: [lukoyanovm@gmail.com](mailto:lukoyanovm@gmail.com)

M.V. Lukoyanov

Nizhny Novgorod State Medical Academy, Nizhny Novgorod, Russian Federation

M.V. Lukoyanov

Higher School of Economics, Nizhny Novgorod, Russian Federation

## 2 Materials and Methods

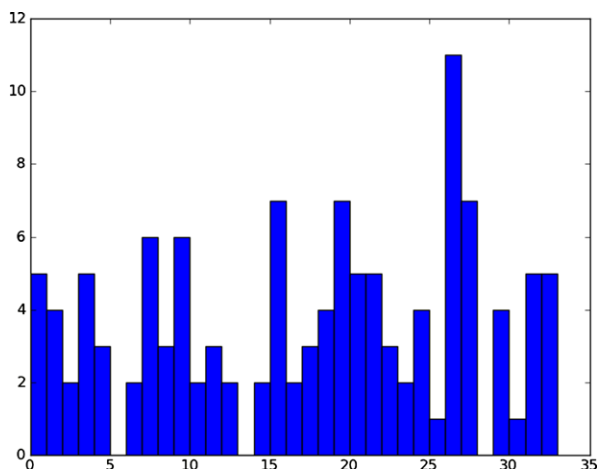
22 volunteers, at age from 18 to 25 took part in the experiments. They were divided into two groups: left-handed (4 men, 5 women) and right-handed (3 men, 10 women). The Edinburgh questionnaire was used to separate subjects into groups [2], as well as standard test of sensorimotor functional asymmetry [3]. EEG was recorded monopolar, ipsilaterally. The electrodes was mounted by the 10–20 system in the following locations: Fp1, Fp2, F3, F4, Fz, C3, C4, Cz, P3, P4, Pz, O1, O2, F7, F8, T3, T4, T5, T6. General referent electrode was placed on the ear lobes. The ground electrode was placed of the forehead. EEG signal was digitally filtered at 0.5–35 Hz. Sampling rate was 500 Hz.

**The procedure of the experiment** The subjects were asked to look at the monitor, without any additional tasks. A white screen was demonstrated to subjects during 30 sec, and then the screen with a different line angle was shown during 60 sec. Each line was demonstrated during 2 sec. Sequence of angles of the lines was pseudorandom and identical for every subjects. 2 sessions of 120 sec. were recorded for each subject. Artifacts of EEG were removed using ICA. Coherence between all pairs of 19 leads (171 pairs) was calculated for each segment of EEG. Coherence calculated follow next formula:

$$C_{xy} = \frac{|P_{xy}|^2}{P_{xx} \times P_{yy}}, \quad (1)$$

where  $P_{xx}$  and  $P_{yy}$  is power spectral density of two signals  $x$  and  $y$  respectively and  $P_{xy}$  is cross power spectral density of  $x$  and  $y$ . For calculation of power spectral density and cross power spectral density, *matplotlib.mlab.psd* and *matplotlib.mlab.csd* functions of matplotlib library for the Python programming language was used. The power spectral density and cross power spectral density was calculated by Welch's average periodogram method. The vectors  $x$  and  $y$  are divided into blocks with 500 samples length. Each block is detrended by the function *matplotlib.pyplot.detrend\_linear* and windowed by the function *numpy.hamming*. Length of the overlap between blocks was 50 %. The  $absolute(fft(block))^2$  of each segment are averaged to compute power spectral density. The product of the direct FFTs of  $x$  and  $y$  was averaged over each segment to compute cross power spectral density. Then coherence was calculated and values for each segment were averaged. The values obtained for the coherence of different frequencies were averaged for the following frequency bands: theta-1 (4–5 Hz), theta-2 (6–7 Hz), alpha-1 (8–9 Hz), alpha-2 (10–12 Hz), beta-1 (13–19 Hz), beta-2 (20–30 Hz). Significant changes of the coherence between the segments with different visual stimulation within each group was determinate by Wilcoxon test (Wilcoxon signed-rank test), for intergroup comparisons—by Mann–Whitney test (Wilcoxon rank-sum test), sold in a package COIN for R programming language (exact distribution was used). Significance of differences was calculated separately for each pair of electrodes. Also for each subject  $p$ -value of Mann–Whitney test between two types of visual stimulation was calculated for not averaged value of coherence. Then Fisher's exact test

**Fig. 1** Numbers of pairs of leads with significant difference between groups in numbers of subjects with significant changes of coherence between two types of visual stimulation.  $X$ -axis is frequency of EEG in Hz and  $Y$ -axis is numbers of pairs of leads with significant difference between groups



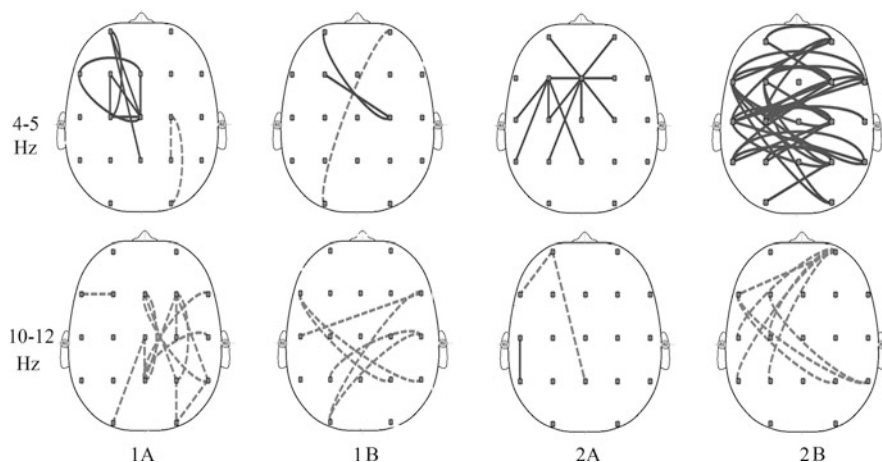
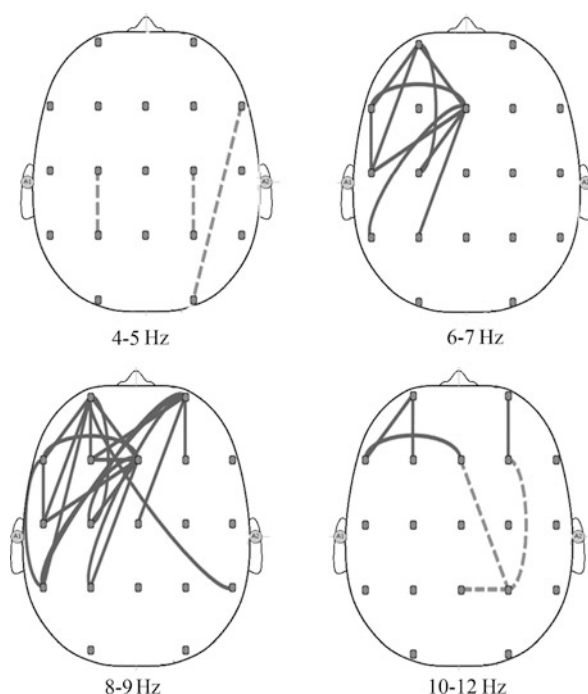
(*scipy.stats.fisher\_exact* function of scipy library for the Python programming language) was used to determinate significance of difference in numbers of subjects with significance changes of coherence between two types of visual stimulation. Level of significant differences was  $p < 0.05$  for all tests.

### 3 Results

Significant difference in numbers of subjects with significance changes of coherence between two types of visual stimulation for all pairs of leads show that difference between groups exhibited inside of standard EEG bands (Fig. 1). There are no differences between groups on the border of standard EEG bands. Coherence in the left hemisphere in group of left-handers vs. right-handers was higher in the theta-2, alfa-1 and beta-bend in frontal, temporal, central and parietal regions during passive observation of white screen (Fig. 2). A similar pattern of differences was observed during passive viewing of lines with different slope. In addition, the left-handers have higher value of coherence in the inter-hemispheric connections in the alpha-1 and beta-2 (in single case in the beta-1) bands in the front regions. In group of left-handers during viewing of lines with different slope vs. white screen values of coherence increased in inter-hemispheric connections in the theta-1 band in almost all leads (Fig. 3). Less marked changes occurred in the decrease of inter-hemispheric connections in the alpha-2 range between temporal and parietal leads of the left hemisphere and the frontal and parietal leads of right hemisphere. Changes in group of right-handers were more localized. As well as in left-handers, theta-band coherence increased, but these changes were less widespread and occurred mainly for intra-hemispheric connections in front of the left hemisphere. In the posterior part of the right hemisphere, there was a decrease of coherence in the alpha-2 and in a few leads in the alpha-1 bands. In the alpha-2 band, coherence decreased between



**Fig. 2** Significant ( $p < 0.05$ ) difference of coherence in group of left-handers vs. right-handers during passive observation of white screen. The *solid lines* are the pairs of leads with higher value of coherence in left-handers, *dashed*—right-handers



**Fig. 3** Significant ( $p < 0.05$ ) changes of coherence in the theta-1 (4–5 Hz) and alpha-2 (10–12 Hz) bands during the passive viewing of lines with different slope vs. white screen in right-handers (1) and left-handers (2). Letters refer to intra-hemispheric (A) and inter-hemispheric (B) connections. The *solid lines* are the pairs of leads with increase of coherence, *dashed*—decrease

hemispheres. It should be noted a tendency to decrease of coherence in theta-1 band in the right hemisphere.

## 4 Discussion

In the present study, we compared the level of EEG coherence and its change under two types of background load in subjects with different hand preference. In our opinion, the most interesting results were obtained in the range of slow-wave activity. Increase of inter-hemispheric interaction has been shown in the left-handers during the passive viewing of visual stimuli in the theta range, while in right-handers, these changes occurred locally in the left hemisphere. In the alpha-2 band, there was a decrease of inter-hemispheric coherence in both groups. In addition, in right-handers intra-hemispheric coherence decreased in the posterior part of the right hemisphere. These results can be analyzed in several ways. First, as the influence of subcortical structures and their interaction with cortical processes. Second, as a reflection of the perception and information processing. Speaking of synchronization mechanisms, it should be noted that activity of the two hemispheres synchronizes on the next two conditions: (1) direct thalamic input to the two hemispheres and (2) reverberating activation between the hemispheres [4]. It should be noted that not only thalamus, but also other subcortical structures play an important role in the genesis of cortical rhythms. For example, it was pointed out that the alpha rhythm likely involves the thalamo-cortical network whereas the theta rhythm likely involves the hippocampal-cortical network. This is accorded with other results indicating that alpha activity is related to semantic memory whereas theta one—to episodic memory [5]. Besides of that, it was noted that the sensory-related regions of the posterior lobe, where sensory information is stored, the prefrontal cortex, where the relevant information held and constantly updated, and the hippocampus are working as a single unit with the theta synchronization [6]. Thus, an increase of coherence in the theta range in our study is a reflection of both cortical interactions and subcortical influences, mainly, of the hippocampus. Activity in the alpha-2 band, apparently reflects thalamic influence, especially in the posterior part in right-handers. At the same time in right-handers changes were locally, mainly intra-hemispheric, which may indicate the competitive relationship between the hemispheres. In the left-handers there is the reverse pattern—active inter-hemispheric interaction, which is consistent with the notion that in the left-handers mutually supportive interactions between the hemispheres are dominant [7]. Local representation of elementary functions in right-handers favors integration of similar units and consequently specialization for behaviors which demand fine sensorimotor control, such as manual skills and speech, whereas a diffuse representation in left-handers may lead to integration of dissimilar units and hence specialization for behaviors requiring multitudinal coordination, such as the various spatial abilities [8]. Changes in the theta and alpha bands are often linked with different cognitive processes and attention [5, 6, 9]. In

addition, there is evidence that episodic memory is associated with a greater hemispheric interaction [10]. Theta range mainly reflects top-down, and alpha one—bottom-up processes [11]. Manifestation of the process widely involved in cognitive functions during passive viewing task in our study may be reflection of their more basic meaning. Recently, it has been proposed that the critical role of the hippocampus is to rapidly, continuously, and obligatorily form associations among disparate elements across space and time, and further, to enable the comparison of internal representations with current perceptual input [12]. In addition, it was suggested that the left hemisphere is forming the stable picture of the world and the right hemisphere is detecting inconsistencies in the picture. It is assumed that the process of correcting the world picture of the left hemisphere requires active inter-hemispheric interaction [13]. The processes described above are reflected in the change of EEG coherence. Thus, compared to lesser skilled shooters, experts engage in less cortico-cortical communication, particularly between left temporal association and motor control regions, which implies decreased involvement of cognition with motor processes [14]. These observations suggest an important role of subcortical structures in conscious human activity, and the level of coherence as a reflection of the degree of involvement of cortical structures in the conscious activity. It can be assumed that the formation of local highly specialized systems in right-handers during the perception lead to less involvement of conscious mechanisms, whereas in the left-handers they play a more important role. The right-handers had mostly local intra-hemispheric changes of EEG coherence, while these changes in the left-handers were inter-hemispheric.

## References

1. Alfimova, M.V., Mel'nikova, T.S., Lapin, I.A.: Ispol'zovanie kogerentnogo analiza EEG i ego reaktivnosti na psikhofiziologicheskie testy pri pervom epizode u bol'nykh shizofreniy [The use of coherent EEG analysis and reactivity to psychophysiological tests in the first episode of schizophrenia]. *Zh. Nevrol. Psikiatr. Im. S.S. Korsakova* **110**(3), 97–102 (2010)
2. Oldfield, R.C.: The assessment and analysis of handedness: the Edinburgh inventory. *Neuropsychologia* (1971). doi:[10.1016/0028-3932\(71\)90067-4](https://doi.org/10.1016/0028-3932(71)90067-4)
3. Nikolaeva, E.I., Yu, B.E.: Sravnenie raznykh sposobov otsenki profilya funktsional'noy sensorimotornoy asimmetrii u doskol'nikov [The comparison of different methods of functional sensorimotor asymmetry profile appritiation for preschool children]. *Asimetriya* **2**(1), 32–39 (2008)
4. Knyazeva, M.G., Kiper, D.C., Vildavski, V.Y., et al.: Visual stimulus-dependent changes in interhemispheric EEG coherence in humans. *J. Neurophysiol.* **82**(6), 3095–3107 (1999)
5. Klimesch, W., Hanslmayr, S., Sauseng, P., et al.: Oscillatory EEG correlates of episodic trace decay. *Cereb. Cortex* **16**(2), 280–290 (2006)
6. Sarnthein, J., Petsche, H., Rappelsberger, P. et al.: Synchronization between prefrontal and posterior association cortex during human working memory. *Proc. Natl. Acad. Sci. USA* **95**(12), 7092–7096 (1998)
7. Zhavoronkova, L.A.: Osobennosti mezhpolutsarnoy asimmetrii EEG pravshey i levshyey kak otrazhenie vzaimodeystvie kory i regulatorynykh sistem mozga [EEG features of inter-hemispheric asymmetry of right-handers and left-handers as a reflection of the interaction of the cortex and the regulatory systems of the brain]. *Dokl. Akad. Nauk SSSR* **375**(5), 696–699 (2000)

8. Semmes, J.: Hemispheric specialization: a possible clue to mechanism. *Neuropsychologia* (1968). doi:[10.1016/0028-3932\(68\)90035-3](https://doi.org/10.1016/0028-3932(68)90035-3)
9. Başar, E., Güntekin, B.: A short review of alpha activity in cognitive processes and in cognitive impairment. *Int. J. Psychophysiol.* **86**(1), 25–38 (2012)
10. Christman, S.D., Butler, M.: Mixed-handedness advantages in episodic memory obtained under conditions of intentional learning extend to incidental learning. *Brain Cogn.* **77**(1), 17–22 (2011)
11. von Stein, A., Sarnthein, J.: Different frequencies for different scales of cortical integration: from local gamma to long range alpha/theta synchronization. *Int. J. Psychophysiol.* **38**(3), 301–313 (2000)
12. Olsen, R.K., Moses, S.N., Riggs, L., Ryan, J.D.: The hippocampus supports multiple cognitive processes through relational binding and comparison. *Front. Human Neurosci.* **6**, 146 (2012)
13. Niebauer, C.L., Garvey, K.: Gödel, Escher and degree of handedness: differences in interhemispheric interaction predict differences in understanding self-reference. *Laterality* **9**(1), 19–34 (2004)
14. Deeny, S.P., Hillman, C.H., Janelle, C.M., Hatfield, B.D.: Cortico-cortical communication and superior performance in skilled marksmen: an EEG coherence analysis. *J. Sport Exerc. Psychol.* **25**(2), 188–204 (2003)

# Speeding up MCS Algorithm for the Maximum Clique Problem with ILS Heuristic and Other Enhancements

Evgeny Maslov, Mikhail Batsyn, and Panos M. Pardalos

**Abstract** In this chapter, we present our enhancements of one of the most efficient exact algorithms for the maximum clique problem—MCS algorithm by Tomita, Sutani, Higashi, Takahashi and Wakatsuki (in Proceedings of WALCOM'10, 2010, pp. 191–203). Our enhancements include: applying ILS heuristic by Andrade, Resende and Werneck (in Heuristics 18:525–547, 2012) to find a high-quality initial solution, fast detection of clique vertices in a set of candidates, better initial coloring, and avoiding dynamic memory allocation. A good initial solution considerably reduces the search tree size due to early pruning of branches related to small cliques. Fast detecting of clique vertices is based on coloring. Whenever a set of candidates contains a vertex adjacent to all candidates, we detect it immediately by its color and add it to the current clique avoiding unnecessary branching. Though dynamic memory allocation allows to minimize memory consumption of the program, it increases the total running time. Our computational experiments show that for dense graphs with a moderate number of vertices (like the majority of DIMACS graphs) it is more efficient to store vertices of a set of candidates and their colors on stack rather than in dynamic memory on all levels of recursion. Our algorithm solves  $p_{\text{hat}}1000-3$  benchmark instance which cannot be solved by the original MCS algorithm. We got speedups of 7, 3000, and 13000 times for `gen400_p0.9_55`, `gen400_p0.9_65`, and `gen400_p0.9_75` instances, correspondingly.

---

E. Maslov (✉) · M. Batsyn · P.M. Pardalos

Laboratory of Algorithms and Technologies for Network Analysis, National Research University Higher School of Economics, 136 Rodionova, Nizhny Novgorod, Russian Federation  
e-mail: [lyriccoder@gmail.com](mailto:lyriccoder@gmail.com)

M. Batsyn

e-mail: [mbatsyn@hse.ru](mailto:mbatsyn@hse.ru)

P.M. Pardalos

e-mail: [pardalos@ufl.edu](mailto:pardalos@ufl.edu)

P.M. Pardalos

Center of Applied Optimization, University of Florida, 401 Weil Hall, P.O. Box 116595, Gainesville, FL 32611-6595, USA

**Keywords** Maximum clique problem · MCS branch-and-bound algorithm · ILS heuristic · Graph coloring

## 1 Introduction

The maximum clique problem refers to the problem of finding a clique (a complete subgraph) with the largest number of vertices in a given graph. It has a lot of practical applications (Bomze et al., 1999 [3]; Du and Pardalos, 1999 [6]; Boginski et al., 2003 [2]).

There exists a lot of exact branch-and-bound algorithms for the solving the maximum clique problem: Bron and Kerbosch (1973) [4], Carraghan and Pardalos (1990) [5], Fahle (2002) [7], Tomita and Seki (2003) [18], Tomita and Kameda (2007) [17], Tomita et al. (2010) [19], Li and Quan (2010) [14, 15] and others. According to the published results for DIMACS graphs MCS algorithm by Tomita et al. [19] and the Max-Sat based algorithms by Li and Quan [14, 15] report the best performance among the existing exact methods known to the authors.

Since the maximum clique problem is NP-hard (Garey and Johnson, 1979, [9]), there also exists a number of heuristic approaches which find solutions of high quality: Kopf and Ruhe (1987) [13], Jerrum (1992) [12], Feo and Resende (1995) [8], Glover and Laguna (1997) [10], Singh and Gupta (2006) [16], Grosso et al. (2008) [11] and others. One of the most successful heuristic algorithms is the iterated local search (ILS) developed by Andrade et al. (2012) [1].

In our approach, we apply ILS heuristic (Andrade et al., 2012 [1]) to obtain an initial solution and speed up the original MCS algorithm. This solution provides a good lower bound and many branches having an upper bound not greater than this lower bound are pruned. Our computational results prove the effectiveness of this approach especially for large dense graphs due to the great reduction in the number of the search tree nodes.

Another enhancement we apply follows the ideas of Fahle (2002) [7] to add the vertices adjacent to all candidates to the current clique immediately without branching. We suggest an algorithm of fast detection of such vertices by means of coloring. The next enhancement is using of a full-fledged greedy coloring at the first node of the search tree without reordering of vertices. Finally, we propose to avoid dynamic memory allocation and use only fast stack memory for candidates and their colors on all steps of the algorithm.

The chapter is organized as follows. In the next section, we describe the maximum clique problem and prove three propositions needed fast detection of clique vertices. Section 3 contains a description of our algorithm. Computational results showing a comparison with the original MCS algorithm are presented in Sect. 4.

## 2 Maximum Clique Problem

Let  $G = (V, E)$  be an undirected graph, where  $V = \{1, 2, \dots, n\}$  is the set of vertices,  $E \subseteq V \times V$  is a set of edges. The adjacency matrix of  $G(V, E)$  is denoted as  $A = (a_{ij})$ , where  $a_{ij} = 1$  if  $(i, j) \in E$  and  $a_{ij} = 0$  if  $(i, j) \notin E$ . A *complementary graph* of  $G(V, E)$  is the graph  $\overline{G}(V, \overline{E})$ , where  $\overline{E} = (V \times V) \setminus E$ . Graph  $G(V, E)$  is *complete* if all its vertices are pairwise adjacent, i.e.  $\forall i, j \in V, (i, j) \in E$ . A *clique*  $C$  is a subset of  $V$  such that all vertices in this subset are pairwise adjacent. A clique which has the maximum size (number of vertices) in a graph is called a *maximum clique*. The number of vertices of a maximum clique in graph  $G(V, E)$  is denoted by  $\omega(G)$ . The maximum clique problem refers to the problem of finding a maximum clique in a given graph.

It is possible to formulate the notion of a *clique* in terms of a chromatic number: a clique is a graph which chromatic number is equal to the number of vertices. It can help to identify a clique by means of coloring (see Proposition 1).

**Definition 1** A reasonable coloring is a sequential coloring which would not use a new color for a vertex if it is possible to color it into one of the already used colors.

**Proposition 1** *If a graph with  $k$  vertices is colored in  $k$  colors by a reasonable coloring, then this graph is complete.*

*Proof* By contradiction: let this graph be incomplete. This means that it has at least two non-adjacent vertices. Let these vertices be  $i$  and  $j$ , and vertex  $i$  is colored before vertex  $j$  by our reasonable coloring. Since  $k$  vertices are colored in  $k$  colors then every vertex has a unique color. But then vertex  $j$  will be colored into the same color as vertex  $i$  because no other vertex is colored in this color and our coloring is reasonable. This contradicts with the condition that  $k$  colors are used.  $\square$

Vertices with higher degree are usually contained in larger cliques. For example, a vertex which is adjacent to all other vertices in a graph is contained in all cliques including the maximum one. Such vertices can be found faster after coloring of a given graph (see Propositions 2 and 3).

**Proposition 2** *If a vertex is adjacent to all vertices of a graph, then it will have a unique color in any coloring of the graph.*

*Proof* By definition, another vertex cannot have the same color in a coloring because it is adjacent to this vertex.  $\square$

**Definition 2** A greedy coloring is a sequential coloring which on every step colors a vertex of a graph in a minimal possible color.

**Proposition 3** *For any greedy coloring if a vertex has a unique color then it is adjacent to all vertices with greater color.*

*Proof* By contradiction: let there be a vertex  $i$  with a unique color  $k$  and a vertex  $j$  with a greater color which is not adjacent to  $i$ . Vertex  $j$  should be colored in color  $k$  or even smaller color because no vertices have color  $k$  except  $i$  which is not adjacent to  $j$  and our coloring always uses a minimal possible color. This contradicts with the condition that  $j$  has a greater color than  $k$ .  $\square$

Following the *df* algorithm (Fahle, 2002 [7]) we check whether there are candidates which are adjacent to all the other candidate vertices and thus should be immediately added to the current clique without any branching. In contrast with the *df* implementation, we can check it much faster using Propositions 2 and 3, because we have to check only vertices which have a unique color after coloring. For such a vertex, we should only check if it is adjacent to all vertices with a smaller color.

Following the MCS algorithm of Tomita et al. (2010) [19] for an upper bound in our algorithm we use the same greedy coloring which assigns the minimal possible color number to every vertex one by one.

### 3 Algorithm Description

The main procedure of our algorithm has the following steps:

1. Find an initial clique with the ILS heuristic (see Andrade et al. (2012) [1] for a detailed description).
2. Perform initial ordering of vertices as follows. Find the vertex with the minimum degree in the graph and put it to the beginning of the current candidates set  $S$ . If several vertices have the same minimum degree then take the vertex which has the minimum support (sum of the vertex neighbors degrees) among them. For several vertices having the same minimum support, ties are broken arbitrarily. Then delete it from the graph and repeat this step until the graph has edges. The remaining set of vertices called  $Rmin$  is put to the beginning of  $S$ .
3. Color the current candidates set  $S$  with a standard greedy sequential coloring (use minimal possible color on each step) from the beginning ( $Rmin$  set) to the end.
4. While the current candidates set  $S$  has unconsidered candidates call the recursive brand-and-bound procedure described below.

The recursive branch-and-bound procedure has the following steps:

1. Consider the last candidate in  $S$ . If its color number plus the size of the current clique is less or equal to the size of the best found clique then:
  - If we are not at the first level of recursion return from recursion because the candidates are ordered by colors and all the remaining candidates have a smaller color number.
  - Otherwise consider the candidate which is previous to this one in  $S$  and repeat this step for it.



2. Add this candidate to the current clique  $Q$ .
3. Remove it from the current candidates set  $S$ .
4. Form a new candidates set  $S'$  taking from set  $S$  only the neighbors of this candidate. We take these neighbors from  $S$  in the initial order obtained on step 2 of the main procedure.
5. Color the new candidates set  $S'$  with standard greedy sequential coloring.
6. If the number of colors is equal to the size of the new candidate set  $S'$ , add all these candidates to the current clique (see Proposition 1). Return from recursion.
7. If there are candidates having a unique color (see Proposition 2), check each of them if it is adjacent to all the candidates with a smaller color (see Proposition 3). If yes, then add such vertices to the current clique and remove them from  $S'$ .
8. Sort the new candidates set  $S'$  in the ascending order with respect to their color numbers.
9. Recursively call this procedure for the new candidates set  $S'$

To store candidate vertices and their colors on every level of the search tree, we allocate stack memory of a predetermined size, enough for any DIMACS graph with the number of vertices from 100 to 1500 and for any possible search tree depth. We suggest to use a greedy coloring on the first step, but do not reorder the vertices by their color numbers as in MCS algorithm. So on the first level we have to keep in the candidates set those vertices for which their color number plus the current clique size is not greater than the currently best clique size.

## 4 Computation Results

We test our algorithm on hard DIMACS instances which need more than 10 minutes to be solved. The computational results are presented in Tables 1 and 2. Table 1 shows the number of the search tree nodes for our algorithm and for MCS algorithm. The last two columns contain the size of the maximum clique and the size of the best clique found by the ILS heuristic applied in our algorithm. We run the ILS heuristic with 100000 scans for all the considered instances except *gen400\_p0.9\_55* and *p\_hat1000-3* for which we use 60 millions scans. The greatest reduction of the search tree size in comparison with MCS algorithm is got for *gen* instances: 60 times reduction for *gen400\_p0.9\_55*, 7000 times for *gen400\_p0.9\_65* and 240000 times for *gen400\_p0.9\_75*.

The comparison of the computational time for our algorithm and MCS algorithm is given in Table 2. The best results are obtained for *gen400\_p0.9\_75* (our algorithm is 13000 times faster for it), *gen400\_p0.9\_65* (3000 times faster), *gen400\_p0.9\_55* (7 times faster), and *p\_hat1000-3* graphs. MCS algorithm is unable to solve *p\_hat1000-3* instance (at least in 10 days) while our algorithm solves it

**Table 1** The number of search tree nodes

Benchmark	Our algorithm	MCS	Maximum clique	Heuristic solution
brock800_1	1095645796	1097174023	23	19
brock800_2	970862419	972110520	24	20
brock800_3	625139200	625234820	25	19
brock800_4	424101537	424176492	26	19
gen400_p0.9_55	55079436	3425049256	55	54
gen400_p0.9_65	822991	6500277298	65	64
gen400_p0.9_75	41445	10140428816	75	75
keller5	10337321299	10339211493	27	27
MANN_a45	219979	221476	345	344
p_hat1000-2	21587044	25209207	46	44
p_hat1000-3	8773710250	–	68	67
p_hat1500-2	607200969	660539819	65	61
p_hat700-2	416003	670369	44	42
p_hat700-3	81631372	98911559	62	60

**Table 2** Running time

Benchmark	ILS	Our algorithm	MCS
brock800_1	167	6482	6337
brock800_2	164	5886	5737
brock800_3	166	3994	3830
brock800_4	166	3009	2849
gen400_p0.9_55	4320	5133	39015
gen400_p0.9_65	8	25	77620
gen400_p0.9_75	7	8	110579
keller5	96	78957	78875
p_hat1000-2	172	360	272
p_hat1000-3	54185	214611	> 1000000
p_hat1500-2	346	10231	11859
p_hat700-3	44	1303	1529

in 3 days. The total time of solving all the considered DIMACS instances is reduced by 75 %.

**Acknowledgements** The authors would like to thank professor Mauricio Resende and his co-authors for the source code of their powerful ILS heuristic. The authors are supported by LATNA Laboratory, National Research University Higher School of Economics (NRU HSE), Russian Federation government grant, ag. 11.G34.31.0057.

## References

1. Andrade, D.V., Resende, M.G.C., Werneck, R.F.: Fast local search for the maximum independent set problem. *J. Heuristics* **18**(4), 525–547 (2012). doi:[10.1007/s10732-012-9196-4](https://doi.org/10.1007/s10732-012-9196-4)
2. Boginski, V., Butenko, S., Pardalos, P.M.: On structural properties of the market graph. In: Nagurney, A. (ed.) *Innovations in Financial and Economic Networks*, pp. 29–45. Edward Elgar, Cheltenham Glos (2003)
3. Bomze, I., Budinich, M., Pardalos, P.M., Pelillo, M.: *Handbook of Combinatorial Optimization*. Kluwer Academic, Dordrecht (1999). Chap. “The maximum clique problem”
4. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM* **16**(9), 575–577 (1973). doi:[10.1145/362342.362367](https://doi.org/10.1145/362342.362367)
5. Carraghan, R., Pardalos, P.M.: An exact algorithm for the maximum clique problem. *Oper. Res. Lett.* **9**(6), 375–382 (1990). doi:[10.1016/0167-6377\(90\)90057-C](https://doi.org/10.1016/0167-6377(90)90057-C)
6. Du, D., Pardalos, P.M.: *Handbook of Combinatorial Optimization, Supplement*, vol. A. Springer, Berlin (1999)
7. Fahle, T.: Simple and fast: improving a Branch-and-Bound algorithm for maximum clique. In: *Proceedings of the 10th Annual European Symposium on Algorithms, ESA '02*, pp. 485–498. Springer, London (2002)
8. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. *J. Glob. Optim.* **6**(2), 109–133 (1995). doi:[10.1007/BF01096763](https://doi.org/10.1007/BF01096763)
9. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York (1979)
10. Glover, F., Laguna, M.: *Tabu Search*. Kluwer Academic, Dordrecht (1997)
11. Grosso, A., Locatelli, M., Pullan, W.: Simple ingredients leading to very efficient heuristics for the maximum clique problem. *J. Heuristics* **14**(6), 587–612 (2008). doi:[10.1007/s10732-007-9055-x](https://doi.org/10.1007/s10732-007-9055-x)
12. Jerrum, M.: Large cliques elude the metropolis process. *Random Struct. Algorithms* **3**(4), 347–359 (1992). doi:[10.1002/rsa.3240030402](https://doi.org/10.1002/rsa.3240030402)
13. Kopf, R., Ruhe, G.: A computational study of the weighted independent set problem for general graphs. *Found. Control Eng.* **12**, 167–180 (1987)
14. Li, C.M., Quan, Z.: Combining graph structure exploitation and propositional reasoning for the maximum clique problem. In: *Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI'10*, vol. 1., Arras, France, pp. 344–351. IEEE Press, New York (2010)
15. Li, C.M., Quan, Z.: An efficient branch-and-bound algorithm based on maxsat for the maximum clique problem. In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI-10*, pp. 128–133. AAAI Press, Atlanta (2010)
16. Singh, A., Gupta, A.K.: A hybrid heuristic for the maximum clique problem. *J. Heuristics* **12**(1–2), 5–22 (2006). doi:[10.1007/s10732-006-3750-x](https://doi.org/10.1007/s10732-006-3750-x)
17. Tomita, E., Kameda, T.: An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *J. Glob. Optim.* **37**(1), 95–111 (2007)
18. Tomita, E., Seki, T.: An efficient branch-and-bound algorithm for finding a maximum clique. In: *Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science, DMTCS'03*, pp. 278–289. Springer, Berlin (2003)
19. Tomita, E., Sutani, Y., Higashi, T., Takahashi, S., Wakatsuki, M.: A simple and faster branch-and-bound algorithm for finding a maximum clique. In: *Proceedings of the 4th International Conference on Algorithms and Computation, WALCOM'10*, pp. 191–203. Springer, Berlin (2010)

# Summary and Semi-average Similarity Criteria for Individual Clusters

**Boris Mirkin**

**Abstract** There exists much prejudice against the within-cluster summary similarity criterion which supposedly leads to collecting all the entities in one cluster. This is not so if the similarity matrix is preprocessed by subtraction of “noise”, of which two ways, the uniform and modularity, are analyzed in the chapter. Another criterion under consideration is the semi-average within-cluster similarity, which manifests more versatile properties. In fact, both types of criteria emerge in relation to the least-squares data approximation approach to clustering, as shown in the chapter. A very simple local optimization algorithm, Add-and-Remove( $S$ ), leads to a sub-optimal cluster satisfying some tightness conditions. Three versions of an iterative extraction approach are considered, leading to a portrayal of the cluster structure of the data. Of these, probably most promising is what is referred to as the injunctive clustering approach. Applications are considered to the analysis of semantics, to integrating different knowledge aspects and consensus clustering.

**Keywords** Summary similarity · Semi-average similarity · Least-squares approximation · Injunctive clustering

## 1 Graph Theoretic Cluster Relevant Concepts

Currently, the most popular format for similarity data is of square matrix  $A = (a_{ij})$  of pair-wise indices  $a_{ij}$  expressing similarity between entities  $i, j \in I$ . The greater the value of  $a_{ij}$ , the greater is the similarity between  $i$  and  $j$ . Intuitively, cluster is a set of highly similar entities that are dissimilar from entities outside of the cluster. Some examples of similarity data are (1) individual judgments of similarity expressed using a fixed range, (2) correlation coefficients between variables or time

---

B. Mirkin (✉)

Division of Applied Mathematics, Higher School of Economics, Moscow, Russian Federation  
e-mail: [bmirkin@hse.ru](mailto:bmirkin@hse.ru)

B. Mirkin

Department of Computer Science and Information Systems, University of London, Birkbeck, UK

**Table 1 Functions:**  
Similarity scores between  
nine elementary functions as  
rated by a high-school  
mathematics teacher

Function	$e^x$	$\ln x$	$1/x$	$1/x^2$	$x^2$	$x^3$	$\sqrt{x}$	$\sqrt[3]{x}$
$\ln x$	7							
$1/x$	1	1						
$1/x^2$	1	1	7					
$x^2$	2	2	2	2				
$x^3$	3	2	1	1	6			
$\sqrt{x}$	2	4	1	2	5	4		
$\sqrt[3]{x}$	2	4	1	1	5	3	5	
$ x $	2	3	1	1	5	2	3	2

series, (3) graphs represented by 1/0-similarity matrices, (4) weighted graphs, or networks, (5) probabilities of common ancestry, especially in proteomics, (6) affinity data obtained by transformation of distances using a Gaussian or another kernel function. Here are two data sets of this type:

1.1 Similarity Between Elementary Functions

There are common knowledge control devices, oral questioning or written test papers: the mark depends on the match between the answers and corresponding knowledge fragments. A few decades ago, an education research team in Russia proposed a different method for knowledge control based on the respondent scoring similarity between the base concepts of the subject tested. The premise is that there exists a structure of semantic relationship among the concepts, which must be acquired by learning; the discrepancy between a student’s personal structure and that one to be acquired may be used as a measure of the student’s knowledge extent (see [14, 24]). Therefore, the student score is determined from the differences between two concept-to-concept similarity matrices: first, that considered to be right, and second, that produced by the respondent.

Table 1 of similarities between 9 elementary algebraic functions has been produced by an expert high-school mathematics teacher. How the differences should be evaluated? Well, over their semantic structures. The semantic structure conventionally is determined with a multidimensional scaling method. In this case, however, the semantic space dimensions obviously corresponded to entity clusters, which brought forward an idea of using clusters only. This brings forward the problem of finding a cluster structure by the similarity matrix.

1.2 Eurovision Song Contest Scoring

Eurovision song contest is an all-European television show at which each participating country presents a song performed by her singer. The performances are eval-

**Table 2 Eurovision scoring:** Each row contains the average score given by the row country to the column country in Eurovision song contests (multiplied by 10)

Country	Az	Be	Bu	Es	Fr	Ge	Gr	Is	It	Ne	Pol	Por	Ro	Ru	Se	Sp	Sw	Ukr	UK
1 Azerbaijan	0	0	0	0	0	0	61	48	0	0	0	0	50	65	0	0	0	90	0
2 Belgium	38	0	0	0	0	39	40	0	0	47	0	0	0	0	0	34	0	0	42
3 Bulgaria	67	0	0	0	0	0	93	0	0	0	0	0	0	48	60	0	0	44	0
4 Estonia	41	0	0	0	0	0	0	0	43	0	0	0	0	88	0	0	0	43	0
5 France	0	37	43	0	0	0	0	56	47	0	0	54	0	0	80	0	0	0	41
6 Germany	0	0	0	0	34	0	37	35	0	0	55	0	0	0	70	0	0	0	42
7 Greece	54	0	80	0	41	0	0	0	0	0	0	0	40	0	80	44	0	38	0
8 Israel	50	0	0	0	0	0	0	0	0	43	0	0	66	74	50	0	0	62	43
9 Italy	0	0	100	0	54	0	0	0	0	0	0	0	120	0	0	0	0	65	52
10 Netherlands	39	46	0	0	0	38	0	45	0	0	0	0	0	0	70	0	0	0	0
11 Poland	84	43	0	39	0	0	0	0	90	0	0	0	0	0	0	0	0	82	0
12 Portugal	0	35	0	0	0	45	0	41	81	0	0	0	52	0	57	42	0	74	43
13 Romania	52	0	0	0	0	0	82	0	60	0	0	0	0	49	80	0	0	35	0
14 Russia	99	0	0	0	0	0	37	36	0	0	0	0	0	0	80	0	0	77	0
15 Serbia	0	0	53	0	0	0	73	0	0	0	0	0	0	44	0	0	0	44	0
16 Spain	0	0	78	0	0	51	45	0	74	0	0	43	79	0	47	0	0	46	0
17 Switzerland	0	0	0	0	44	0	0	42	47	0	0	0	0	0	106	41	0	0	41
18 Ukraine	111	0	0	0	0	0	0	0	0	0	60	0	0	98	90	0	0	0	0
19 UK	0	0	0	36	0	39	38	0	0	0	0	0	0	0	37	0	0	0	0

uated by each of the participating countries separately, so that each country selects eleven best performances and gives them scores in the order of preference. The winner is that country whose song received the maximum score combined. The participants frequently suspect that voting might be led by ethnic and cultural relations rather than by the quality of performance.

Table 2 presents a part of the whole picture using data of only 19 European countries, out of 46, listed in its left column. The table, including the sample of countries, has been compiled by the author using scoring data publicly available at <http://www.esccstats.com/> (visited 28/2/2013). Its entries are the average scores given by each country to its 10 top choices at all the Eurovision song contests (up to and including year 2011). I tried to make a representative sample of less than 20 entities so that the reader could see the data with a naked eye. Each row of the table corresponds to one of the nineteen countries, and assigns a nonzero score to those of the other eighteen that have been among the 10 best choices. For the sake of convenience, each score has been multiplied by 10 so that all scores are expressed with integers.

The cluster structure of the table should quantify to what extent the gossip of the effects of cultural and ethnic links on voting is justified, because song and perfor-

mance “quality” may be considered random from year to year, so that in the ideal case when no cultural preferences are involved at evaluations, the similarity matrix should be of a random structure too.

As one can see both similarity matrices are non-symmetric, which is not that important with the clustering criteria used in this chapter.

Formal thinking have been applied to similarity clusters rather early, in graph theory, much before the need in clustering has been recognized as a general problem. A graph may be thought of as a flat 1/0 similarity matrix, its nodes corresponding to row/columns and edges/arcs to entries with ones corresponding to edges. Cluster related graph-theoretic concepts include: (a) *connected component* (a maximal subset of nodes in which there is a path connecting any pair of nodes), (b) *bi-component* (a maximal subset of nodes in which each pair of nodes belongs to a cycle), and (c) *clique* (a maximal subset of nodes in which each pair of nodes is connected by an edge). These remain much relevant in various contexts. For example, define  $S \subset I$  to be a strong cluster if for any  $i, j \in S$  there exists  $k \in I - S$  such that  $a_{ij} > a_{ik}$ . It is not difficult to prove that  $S$  is a strong cluster if and only if  $S$  is a clique for some threshold graph. A threshold graph is defined by a real  $t$  so that  $(i, j)$  is its arc if and only if  $a_{ij} > t$ . Even more relevant is a more recent concept of the maximum density subgraph [6]. The density  $g(S)$  of a subgraph  $S \subset I$  is the ratio of the number of edges in  $S$  to the number of elements  $|S|$ . For an edge weighted graph with weights specified by the matrix  $A = (a_{ij})$ , the density of a subgraph on  $S \subseteq I$   $g(S)$  is defined by the *Rayleigh quotient*  $s^T A s / s^T s$ , where  $s = (s_i)$  is the characteristic vector of  $S$ , viz.  $s_i = 1$  if  $i \in S$  and  $s_i = 0$  otherwise. A subgraph of maximum density represents a cluster. After removing such a cluster from the graph, a maximum density subgraph of the remaining graph can be found. This may be repeated until no “significant” clusters remain. Such an incomplete clustering procedure is natural for many types of data, including protein interaction networks. However, to our knowledge, this method has never been applied to such problems, probably because it involves rather extensive computations. A heuristic analogue can be found in [2]. The maximum density subgraph problem is of interest because it is a reasonable relaxation of the maximum clique problem and fits well into data recovery clustering (see Sect. 4.2). The maximum value of the Rayleigh quotient of a symmetric matrix over any real vector  $s$  is equal to the maximum eigenvalue and is attained at an eigenvector corresponding to this eigenvalue. This gives rise to the so-called *spectral clustering*, a method of clustering based on first finding a maximum eigenvector  $s^*$  and then defining the spectral cluster by  $s_i = 1$  if  $s_i^* > t$  and  $s_i = 0$  otherwise, for some threshold  $t$ . This method may have computational advantages when  $A$  is sparse. Unfortunately, it does not necessarily produce an optimal cluster, but in practice it works well (see also [28]).

## 2 Within Cluster Summary Criterion

### 2.1 Uniformly Shifting Similarities

For any subset  $S \subseteq I$ , maximizing the sum of within- $S$  similarities

$$f(S) = \sum_{i,j \in S} a_{ij} \quad (1)$$

seems a perfect criterion for clustering. It is simple and intuitive. The greater the total within cluster similarity  $f(S)$ , the tighter the cluster.

There is an issue though. If all the similarities are nonnegative, as it usually happens (see data in Tables 1 and 2), the function  $f(S)$  can only increase if any other entities are added to  $S$  so that the maximum  $f(S)$  is reached at  $S = I$ , the universal cluster. This is why the criterion has been applied only to the situations at which the size of the cluster is pre-specified or, in the case of partitioning, the distribution of entities over clusters is pre-specified, say, by restricting admissible portions to those consisting of equal-sized clusters only (as in Kupershtoch and Mirkin [10], see also Kernighan and Lin [9]).

Yet the criterion can be saved if applied to unrestricted similarity values. That is, the criterion does work if the similarity data is preprocessed to admit negative values. An intuitively reasonable way for doing that is by subtracting some “background noise” from the similarity data.

Specifically, Kupershtoch, Trofimov and Mirkin [11] proposed subtraction of a constant value  $\pi$  from all the similarity values so that criterion (1) becomes

$$f(S, \pi) = \sum_{i,j \in S} (a_{ij} - \pi) = \sum_{i,j \in S} a_{ij} - \pi |S|^{sq} \quad (2)$$

where  $|S|^{sq}$  denotes either  $|S|(|S| - 1)$  if the diagonal entries  $a_{ii}$  are absent from the criterion so that only  $i \neq j$  are taken in the sum, or  $|S|^2$ , otherwise. The last expression in (2) is easily obtained with little arithmetic.

The value of similarity shift  $\pi$  can be considered a “soft threshold” so that  $i$  and  $j$  should be put in the same cluster if  $a_{ij} > \pi$  and rather not, otherwise. Usually the user can specify such a threshold value depending on the nature of data and clustering goal. In fact, this also reflects the extent of desired granulation of clusters, that is usually introduced with a less intuitive parameter, the cluster size. This is warranted by the following property.

**Statement 1** *The optimal cluster size according to criterion (2) can only decrease when  $\pi$  grows.*

*Proof* Let us assume that, on the contrary, the optimal  $S_1$  at  $\pi = \pi_1$  and optimal  $S_2$  at  $\pi = \pi_2$  are such that  $|S_1| > |S_2|$  and  $\pi_1 > \pi_2$ . Because of the optimality, two obvious inequalities:  $f(S_1, \pi_1) - f(S_2, \pi_1) \geq 0$  and  $f(S_2, \pi_2) - f(S_1, \pi_2) \geq$



0. Summing these two inequalities together and using (2), we obtain, after little arithmetic, that  $\pi_1|S_2|^{sq} - \pi_1|S_1|^{sq} + \pi_2|S_1|^{sq} - \pi_2|S_2|^{sq} = (\pi_1 - \pi_2)(|S_2|^{sq} - |S_1|^{sq}) < 0$ , because of the assumption. This clearly contradicts the condition that the sum is not negative, and proves the statement.  $\square$

A useful property is that the similarity matrix, which may be not symmetric originally, can be equivalently transformed to a symmetric matrix by summing it with its transpose  $A'$ .

**Statement 2** *A cluster  $S$  optimizes criterion (1) over similarity matrix  $A$  if and only if  $S$  optimizes it over symmetric similarity matrix  $A + A'$ .*

*Proof* Let us take indicator vector  $s$  for subset  $S$  so that  $s_i = 1$  if  $i \in S$  and  $s_i = 0$ , otherwise. Then criterion (1) can be equivalently rewritten as  $f(S) = \sum_{i \in S} \sum_{j \in S} a_{ij} = \sum_i \sum_j a_{ij} s_i s_j$ . The proof follows from the fact that  $s_i s_j = s_j s_i$ .  $\square$

The statement allows us to symmetrize similarities beforehand by putting  $a_{ij} + a_{ji}$  for every  $a_{ij}$ . Therefore, from now on, let us assume that  $A$  is symmetric.

One more property of the criterion is that it leads to provably tight clusters. Let us refer to cluster  $S$  as suboptimal if, for any entity  $i$ , the value of criterion (2) can only decrease if  $i$  changes its state in respect to  $S$ . Entity  $i$  changes its state in respect to  $S$  if it is added to  $S$ , in the case that  $i \notin S$ , or removed from  $S$  if  $i \in S$ .

**Statement 3** *If  $S$  is a suboptimal cluster, then the average similarity  $a(i, S)$  of  $i$  with other entities in  $S$  is greater than  $\pi$  if  $i \in S$ , or less than  $\pi$  if  $i \notin S$ .*

*Proof* Let us assume that  $k \in S$  for a suboptimal  $S$ , but  $a(k, S) < \pi$ . Let us consider difference  $f(S, \pi) - f(S - k, \pi)$  where  $S - k$  is what remains of set  $S$  after  $k$  is removed. For the sake of simplicity, assume that diagonal entries  $(i, i)$  are absent from the sum in (2). Then it is easy to see that  $f(S, \pi) - f(S - k, \pi) = -2 \sum_{i \in S} (a_{ik} - \pi)$  assuming that  $A$  is symmetric. Since this is non-negative according to suboptimality of  $S$ , the following inequality holds:  $\sum_{i \in S} a_{ik} \geq (|S| - 1)\pi$ . Dividing this over  $(|S| - 1)$ , we obtain  $a(k, S) \geq \pi$ , which proves one part of the statement. The other part, for  $k \notin S$ , can be proven similarly.  $\square$

The proof involves a simple formula relating the values of criterion  $f(S)$  in (1) (criterion (2) coincides with (1) if the similarity matrix is pre-processed by subtracting  $\pi$  from all its entries) at two sets differing by just one entity that is present in one of the sets and absent in the other. To make a universal expression for the formula, let us use vector  $z = 2s - 1$  one-to-one relating to  $S \subset I$  so that  $z_i = 1$  if  $i \in S$  and  $z_i = -1$  if  $i \notin S$ . Then change of state of entity  $k$  with respect to  $S$  is expressed as change of the sign of  $z_k$ . Therefore, the change of the criterion value is equal to

$$\Delta(S, k) = f(S \pm k) - f(S) = -2z_k \sum_{i \in S} a_{ik}, \quad (3)$$

under the assumption that the diagonal similarities  $a_{ij}$  are not considered and  $z_k$  in (3) corresponds to  $S$ , that is, taken before the change of sign.

Of course, the problem of finding an optimal  $S$  over a matrix  $A$  with possibly negative entries is NP-hard. A locally-optimal improvement algorithm starting from any  $S \subseteq I$  can be formulated as follows [12, 13]:

### Summary Criterion Add-and-Remove( $S$ )

Input: matrix  $A = (a_{ij})$  pre-normalized, subset  $S$ . Output: suboptimal cluster  $T_S$  and value of the summary criterion  $f(T_S)$ .

1. **Initialization.** Set  $N$ -dimensional  $z$  so that  $z_i = 1$  if  $i \in S$  and  $z_i = -1$ , otherwise. Set the summary similarity equal to  $f(S)$ .

2. **General step.** For each entity  $k \in I$ , compute the value  $c_k = \Delta(S, k)$  according to (3) and find  $k^*$  maximizing it.

3. **Test.** If  $c_{k^*} > 0$ , change the sign of  $z_{k^*}$  in vector  $z$ , after which recalculate the summary criterion by adding  $c_{k^*}$  to it, and go to 2. Otherwise, go to 4. (In the case of massive data, computing the differences in (3) can be costly. Therefore, a vector of these values  $c = (c_k)$  should be maintained and dynamically changed after each addition/removal step.)

4. **Output.** Define  $T_S = \{i | z_i = 1\}$  and output it along with the summary criterion value.

The suboptimality of  $T_S$  and, therefore its tightness in the sense of the statement above is warranted by the step 2 of the algorithm.

Algorithm SC AddRem( $S$ ) utilizes no ad hoc parameters, except for the  $\pi$  of course, so the cluster sizes are determined by the preprocessing steps. Three obvious choices for the starting  $S$  are: (a)  $S = I$ , (b)  $S = \{i, j\}$  such that  $a_{ij}$  is the maximum in  $A$ , and (c)  $S = \{i\}$  for any  $i \in I$ . In the case (c), running a loop over all  $i \in I$  will produce as many different clusters  $T_i$ ; the structure of their overlaps gives a portrayal of the cluster structure in matrix  $A$ . One more strategy would be multiple runs of the algorithm starting from random  $S$ .

The algorithm CAST [3], popular in bioinformatics, is a version of this algorithm, in which  $\Delta(S, k)$  is reformulated as  $\sum_{j \in S} a_{ij} - \pi |S|$  and  $\sum_{j \in S} a_{ij}$  is referred to as the affinity of  $i$  to  $S$ .

Let us apply the algorithm to the Eurovision data made symmetric by summing the matrix with its transpose, then subtracting the average non-diagonal value  $\bar{a} = 35.7193$  and, afterwards, zeroing all the diagonal entries. Let us take the maximum of the final matrix  $A$ , that is,  $a_{1,18} = 165.2807$ . Therefore, let us start with the corresponding cluster  $\{1, 18\} = \{\text{Azerbaijan, Ukraine}\}$ . Then the following entities are added one-by-one because of positive summary similarities: 14. Russia, 267.6; 8. Israel, 162.8; 15. Serbia, 165.1; 7. Greece, 164.4; 13. Romania, 239.7; 3. Bulgaria, 195.0 (the summary similarity to the cluster follows the country name). So far, the maximum of the summary similarity of  $S$  with remaining entities has been the maximum of the entire similarity matrix. From now on, this is not so. The next entity to join in is 9. Italy with the summary similarity 59.2, which is somewhat smaller than the maximum of similarities 65.3, between Italy and France.

This makes a difference. If a partitional algorithm is run, making clusters in parallel, with the summary similarity criterion, then it would build another cluster simultaneously. The other cluster(s) would compete with  $S$  in attracting entities so that further steps could have been impossible. This is exactly the case being reported. Instead of joining  $S$ , Italy would have started a different cluster altogether. Indeed, when run in parallel, say, with agglomeration steps, the summary similarity criterion leads to one more meaningful cluster  $T = \{\text{France, Germany, Italy, Portugal, UK}\}$ .

As we build a single suboptimal cluster for the summary similarity criterion, Italy goes in, after which one more entity with a positive summary similarity to  $S$ , 32.8, Portugal is added to  $S$ . Now the computation halts, because each of the yet unclustered entities has a negative summary similarity to  $S$ . It should be pointed out that the cluster  $S$  is not exactly homogeneous: it contains East-European members somewhat attenuated by a few Mediterranean countries.

## 2.2 Subtraction of Random Interactions

Let us assign each entity  $i \in I$  with a probability of interaction with other entities, equal to the proportion of the summary similarity in  $i$ -th row in the total sum of the similarity values. Then random interactions between two entities will occur with the probability equal to the product of their respective probabilities. Under this interpretation, the random interactions should be subtracted from the similarity values to clear up the “nonrandom part of the total similarity structure”.

The summary criterion  $f(S)$  at the similarities pre-processed in this way is but what is referred to as the modularity of  $S$ . The concept of modularity as a clustering criterion was introduced in Newman and Girvan [22], Newman [21], who use it for partitioning, not for individual cluster finding.

All the contents of the previous section holds for the modularity criterion except Statements 1 and 2. The former is irrelevant because there is nothing variant in the modularity transformation; one may explore, though, putting a changeable factor to the random interactions. The latter should be reformulated by using the so-called modularity attraction (see detail in [15], p. 327).

Multiple runs of Add-and-Remove( $i$ ) at different starting points  $i \in I$  allow to (a) find a better cluster  $S$  maximizing the summary similarity criterion over the runs, and (b) explore the cluster structure of the dataset by analyzing both differing and overlapping clusters.

Let us apply this approach to the Eurovision matrix. To do that, we compute the sums of the rows, the sum of the columns, and the total. Then we subtract from every entry  $a_{ij}$  the product of the sums of the corresponding row and column related to the total,  $a_{i+}a_{+j}/a_{++}$ . In the resulting matrix, all the diagonal entries are made zero, after which the matrix is summed with its transpose to make it symmetric. The summary similarity criterion of subset  $S$  for this matrix is what is referred to as the modularity of  $S$  (doubled, because of the symmetrical summation) [21].

Let us build a cluster starting from  $S = \{1\}$ . Once again the maximum positive similarity to 1 is 18, at 127.7. Therefore, we have now  $S = \{\text{Azerbaijan, Ukraine}\}$ .

Russia, number 14, has the maximum positive summary similarity to this, 215.7, which makes  $S = \{\text{Azerbaijan, Ukraine, Russia}\}$ . But then the order of nearest entities changes (from the case of uniform similarity criterion). Next joining is 11. Poland at 107.5; 4. Estonia at 127.5; 8. Israel at 66.4; 10. The Netherlands at 9.0; and 2. Belgium at 22.8. Here the process stops—all the other entities have negative summary similarity to the final  $S$ . At no step was any need to remove an entity; all the summary within cluster similarities have been positive. This, probably, is even more controversial agglomeration than that found at the uniformly preprocessed data. This is probably because the row and column totals here highly affect the result of data preprocessing so that even relatively small similarity values can remain positive if the totals are small enough.

There exists even more powerful transformation of the similarity data involving the products of the row and column totals, the so-called pseudo-inverse Laplace transformation (see, for example, [16, 20]), which frequently somewhat sharpens the structure hidden in data, but not always, unfortunately.

### 3 Semi-average Criterion

Another seemingly natural individual clustering criterion is the average similarity

$$a(S) = \frac{\sum_{i,j \in S} a_{ij}}{|S|(|S| - 1)} \quad (4)$$

This definition relates to the case at which no diagonal elements of  $A$  are involved. In spite of a rather long history of using this criterion (see, for instance, [15, 16]), this case has never been explicitly analyzed; all the derivations referred to the situation of the diagonal entries present so that the denominator is  $|S|^2$ , leading to simpler formulas. Unfortunately, this criterion cannot work for clustering because its maximum is reached at a pair  $\{i, j\}$  at which  $a_{ij}$  is the maximum value in  $A$ —this is the average similarity in this set. Indeed, further addition of other entities can only decrease the average similarity.

As a remedy,  $a(S)$  can be multiplied by  $|S| - 1$  so that the decrease in  $a(S)$  can be compensated by the increase in the cardinality  $|S|$ :

$$b(S) = \frac{\sum_{i,j \in S} a_{ij}}{|S|} = (|S| - 1)a(S) \quad (5)$$

We refer to this as the semi-average criterion. Consider the change of (5) when entity  $k$  changes its state over  $S$  (again, entries  $a_{ii}$  are considered non-existent). Once again, vector  $z = (z_i)$  is invoked, with  $z_i = 1$ , if  $i \in S$ , and  $z_i = -1$ , otherwise.

**Statement 4** *The change of  $b(S)$  when entity  $k$  changes its state with respect to non-singleton  $S$  is*

$$b(S \pm k) - b(S) = z_k \left[ (|S| + z_k)a(S) - 2(|S| + (1 + z_k)/2)a(k, S) \right] / (|S| + 1) \quad (6)$$

where  $a(k, S)$  is the average similarity between entity  $k$  and entities in  $S$ , and  $a(S)$ , the average within- $S$  similarity (4).

*Proof* Let  $k \in S$  so that  $z_k = 1$ . Then the difference is

$$\begin{aligned} b(S - k) - b(S) &= \left[ \sum_{i,j \in S} a_{ij} - 2 \sum_{i \in S} a_{ik} \right] / (|S| - 1) - \sum_{i,j \in S} a_{ij} / |S| \\ &= \left[ |S| \sum_{i,j \in S} a_{ij} - 2|S| \sum_{i \in S} a_{ik} - (|S| - 1) \sum_{i,j \in S} a_{ij} \right] / (|S|(|S| - 1)) \\ &= \left[ \sum_{i,j \in S} a_{ij} - 2|S| \sum_{i \in S} a_{ik} \right] / (|S|(|S| - 1)) = a(S) - 2a(k, S) \end{aligned}$$

The only place in these rather dull derivations that probably deserves a comment is that the average  $a(k, S)$  is computed over  $|S| - 1$  entities, not  $|S|$  because  $k \in S$  and  $a_{kk}$  is not taken into account. The final expression is exactly (6) at  $z_k = 1$ .

Assume now that  $k \notin S$  and take the difference

$$\begin{aligned} b(S + k) - b(S) &= \left[ \sum_{i,j \in S} a_{ij} + 2 \sum_{i \in S} a_{ik} \right] / (|S| + 1) - \sum_{i,j \in S} a_{ij} / |S| \\ &= \left[ |S| \sum_{i,j \in S} a_{ij} + 2|S| \sum_{i \in S} a_{ik} - (|S| + 1) \sum_{i,j \in S} a_{ij} \right] / (|S|(|S| + 1)) \\ &= \left[ 2|S| \sum_{i \in S} a_{ik} - \sum_{i,j \in S} a_{ij} \right] / (|S|(|S| + 1)) \\ &= [2|S|a(k, S) - (|S| - 1)a(S)] / (|S| + 1) \end{aligned}$$

It is not difficult to see that the final expression corresponds to (6) at  $z_k = -1$ . This completes the proof.  $\square$

Let us refer to expression

$$\alpha(k, S) = a(k, S) - a(S)/2 \quad (7)$$

as the attraction of entity  $k$  to subset  $S$ . A tight cluster  $S$  should have all the attractions of its elements positive and attractions of external entities negative.

This is exactly the case for clusters that are suboptimal over the semi-average criterion.

**Statement 5** *If a subset  $S \subset I$  is suboptimal over criterion  $b(S)$  in (5), then for each entity  $k \in I$  its attraction to  $S$  is not negative if  $k \in S$  and not positive if  $k \notin S$ .*

The proof easily follows from the fact that all the differences (6) must be not positive for a suboptimal  $S$ . If, for example,  $k \in S$ , that is,  $z_k = 1$ , then  $b(S - k) - b(S) = a(S) - 2a(k, S) \leq 0$  and, therefore,  $-2\alpha(k, S) \leq 0$ .

Let us note that the statement much resembles that for the summary similarity criterion at threshold  $\pi = a(S)/2$ . The difference is that  $\pi$  is an expert-given value, whereas half the inner average similarity  $a(S)/2$  is determined by cluster itself. This is not just a chance co-occurrence. The next section will show more general criterion of which these two are just part.

Of course, algorithm Semi-average criterion Add-and-Remove( $S$ ) works exactly as defined before except that this time the change of the criterion is computed using formula (6).

As an example, let us apply the Add-and-Remove algorithm for finding a suboptimal cluster of Eurovision data for the Semi-average criterion, starting again from entity 1. Azerbaijan. Matrix  $A$  is preprocessed by summing it with its transpose, subtracting the average of its non-diagonal entries, and by zeroing the diagonal. The algorithm produces cluster  $S = \{1, 3, 7, 14, 15, 18\}$ , that consists of countries: Azerbaijan, Bulgaria, Greece, Russia, Serbia, Ukraine. This looks a culturally cohesive cluster: all four Slav Christian Orthodox countries plus Greece, which is Christian Orthodox, and Azerbaijan, a former Soviet republic. Take a look at the process: first comes 18. Ukraine, then 14. Russia, then 8. Israel. Then, one by one, 15. Serbia, 7. Greece, 13. Romania, 3. Bulgaria are added. After this the attractions of 8. Israel and 13. Romania become negative, and they are removed one by one. Both the process and result show a relative versatility of the semi-average criterion.

## 4 Approximation Models for Summary and Semi-average Criteria

### 4.1 Median in Hamming Distance Space

Consider the set of all binary relations on  $I$  that are subsets of Cartesian product  $I \times I$ , that is set of all ordered pairs  $(a, b)$ ,  $a, b \in I$ . Among them those one-to-one corresponding to subsets  $S \subseteq I$  are of format  $\tau_S = S \times S$ . Converted to the format of  $|I| \times |I|$  binary matrices, a binary relation  $\rho \in I \times I$  is represented by matrix  $r = (r_{ij})$  in which  $r_{ij} = 1$  if  $(i, j) \in \rho$  and  $r_{ij} = 0$ , otherwise. A binary matrix  $t_S$  corresponding to the “square” binary relation  $\tau_S$  corresponding to subset  $S$  has a clear-cut block structure: it is all zero, except for  $t_{ij} = 1$  if and only if both  $i, j \in S$ . Consider Hamming distance between such binary matrices,  $d(r, s) = \sum_{i,j \in I} |r_{ij} - s_{ij}|$ , which is in fact what is referred to as the city-block distance or, in the binary case, the squared Euclidean distance between  $r, s$  as  $|I| \times |I|$ -dimensional vectors [12]. Given  $n$  binary relations on  $I$ ,  $r^1, r^2, \dots, r^n$ , a median subset  $S$  is defined as corresponding to a “clear-cut block” matrix  $s = (s_{ij})$  minimizing the summary distance  $\sum_{m=1}^n d(s, r^k)$ . Define consensus matrix  $A = \sum_{k=1}^n r^k$  so that  $a_{ij}$  is the number of those among the given relations that relate to  $(i, j)$ , that is,  $r_{ij}^k = 1$ .

Then the problem of finding a median is of finding a cluster according to the uniform summary similarity criterion.

**Statement 6** *Subset  $S$  is a median if and only if it maximizes the uniform summary similarity criterion (2) at threshold value  $\pi = n/2$ .*

The proof rather obviously follows from the definition and the fact that  $|u - v| = (u - v)^2$  for 1/0 variables  $u, v$ .

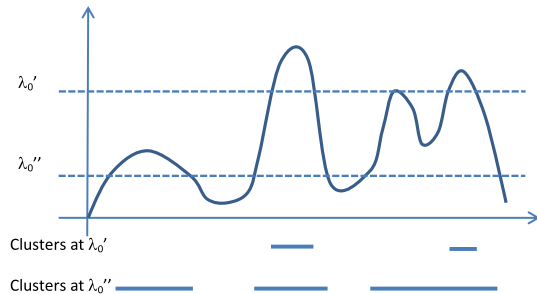
In a specific case of  $n = 1$  the statement can be interpreted as this. Consider this problem: given an ordinary graph, transform it into a complete subgraph (with all outside nodes being isolated) by adding or removing edges so that the number of changes is minimum. This problem is equivalent to the following. Consider a  $1/-1$   $|I| \times |I|$  matrix with its entries being one, if the corresponding edge is in the graph, and minus one, if not. Then finding a cluster  $S$  maximizing  $f(S)$  in (1) over this matrix is the same problem. Of course, this is a hard problem, equivalent to that of finding a complete subgraph with the maximum number of nodes.

## 4.2 Least-Squares Approximation

The idea is to find such a subset  $S \subseteq I$  that its binary matrix  $s = (s_{ij})$  approximates a given symmetric similarity matrix  $A$  as well as possible. To accommodate for the difference in the unit of measurement of the similarity as well as for its zero point, matrix  $s$  should be also supplied with (adjustable) scale shift and rescaling coefficients, say  $\lambda$  and  $\mu$ . That would mean that the approximation is sought in the set of all binary  $\lambda + \mu / \mu$  matrices  $\lambda s + \mu$  with  $\lambda > 0$ . Unfortunately, such an approximation, at least when follows the least squares approach, would have little value as a tool for producing a cluster, because the optimal values for  $\lambda$  and  $\mu$  would not separate the optimal  $S$  from the rest [13]. This is why from about 1995, this author uses only one parameter  $\lambda$ , change of the unit of measurement, in formulating approximation problems in clustering. The issue of adjustment of similarity zero point, in such a setting, is moved out of the modeling stage to the data preprocessing stage. Basically, this amounts to the need in subtraction of a similarity shift value before doing data analysis. Choice of the similarity shift value may affect the clustering results, which can be of advantage in contrasting within—and between—cluster similarities. Figure 1 demonstrates the effect of changing a positive similarity  $a_{ij}$  to  $a'_{ij} = a_{ij} - \lambda_0$  for  $\lambda_0 > 0$ ; small similarities  $a_{ij} < \lambda_0$  are transformed into negative similarities  $a'_{ij}$ .

Therefore, in the remainder of this section, it is assumed that a similarity shift value has been subtracted from all the similarity entries. Another assumption, for the sake of simplicity, is that the diagonal entries  $a_{ii}$  are all zero (after the preprocessing step). One more simplifying assumption is that the subset  $S \subseteq I$  now is presented with a binary vector rather than binary matrix. From now on,  $S$  is represented by a vector  $s = (s_i)$  such that  $s_i = 1$  if  $i \in S$  and  $s_i = 0$ , otherwise.

**Fig. 1** Patterns of clustering depending on the subtracted similarity shift  $\lambda_0$



Therefore, our approximation model is

$$a_{ij} = \lambda s_i s_j + e_{ij} \quad (8)$$

where  $a_{ij}$  are the preprocessed similarity values,  $s = (s_i)$  is the unknown cluster belongingness vector and  $\lambda$ , the rescaling value, also referred to as the cluster intensity value. To fit the model (8), various criteria can be utilized. Only the least squares criterion  $L^2 = \sum_{i,j \in I} e_{ij}^2$  is considered here.

#### 4.2.1 Pre-specified Intensity

We first consider the case in which the intensity  $\lambda$  of the cluster to be found is pre-specified. Remembering that  $s_i^2 = s_i$  for any 0/1 variable  $s_i$ , the least squares criterion can be expressed as

$$L^2(S, \lambda) = \sum_{i,j \in I} (a_{ij} - \lambda s_i s_j)^2 = \sum_{i,j \in I} a_{ij}^2 - 2\lambda \sum_{i,j \in I} (a_{ij} - \lambda/2) s_i s_j \quad (9)$$

Since  $\sum_{i,j} a_{ij}^2$  is constant, for  $\lambda > 0$ , minimizing (9) is equivalent to maximizing the summary within-cluster similarity after subtracting the threshold value  $\pi = \lambda/2$ , i.e.,

$$f(S, \pi) = \sum_{i,j \in I} (a_{ij} - \pi) s_i s_j = \sum_{i,j \in S} (a_{ij} - \pi) \quad (10)$$

This is exactly the summary similarity criterion considered above in Sect. 2.1.

#### 4.2.2 Optimal Intensity

When  $\lambda$  in (9) is not fixed but can be chosen to further minimize the criterion, it is easy to prove that the optimal  $\lambda$  is equal to the average within-cluster similarity:

$$\lambda = a(S) = s^T A s / [s^T s]^2 \quad (11)$$

The second equation follows from the fact that  $s^T s = |S|$ .



By putting this equation in the least-squares criterion (9), it is not difficult to derive a Pythagorean decomposition:

$$L^2(S) = (A, A) - [s^T A s / s^T s]^2 \quad (12)$$

where  $A$  plays the role of hypotenuse. This decomposition implies that the optimal cluster  $S$  is a maximizer of a criterion depending on  $S$  only:

$$g^2(S) = [s^T A s / s^T s]^2 = a^2(S)|S|^2 \quad (13)$$

According to (13), the maximum of  $g^2(S)$  may correspond to either positive or negative value of  $a(S)$ . The latter case may emerge when the preliminarily subtracted similarity shift is large and corresponds to  $S$  being the so-called *anti-cluster* [14]. An anti-cluster consists of entities that all are mutually far away from each other. Therefore, we skip this latter case but rather focus on maximizing (13) only for positive  $a(S)$ . This is equivalent to maximizing its square root, that is the Rayleigh quotient,

$$g(S) = s^T A s / s^T s = a(S)|S| \quad (14)$$

This criterion is a form of the semi-average clustering criterion considered in Sect. 3.

Another form would be obtained for the case at which similarities of entities to themselves,  $a_{ii}$ , are not defined. In this case, the criterion (13) would be  $a^2(S)|S|(|S| - 1)$  so that  $g(S)$  would involve the square root  $\sqrt{|S|(|S| - 1)}$ , thus leading to a bit more complicated formulas in the corresponding local optimization algorithm Add-and-Remove.

Therefore, one can see that both heuristic criteria considered in the beginning are related to the least-squares approximation criterion (9). At a pre-specified  $\lambda$ , the summary similarity clustering criterion with the uniformly subtracted  $\pi = \lambda/2$  is obtained. At the optimal  $\lambda$ , the semi-average similarity clustering criterion emerges. But in fact it is more than just the criterion. Because of the data scatter decomposition in (12), its square,  $g^2(S)$  expresses the share of the similarity data scatter  $(A, A) = \sum_{i,j} a_{ij}^2$  “explained”, or better to say, “taken into account” by the cluster  $S$  and its intensity  $\lambda = a(S)$  found by (locally) optimizing  $g(S)$ . This allows one to judge how well the cluster reflects the structure of  $A$ .

The least-squares criterion  $L^2(S, \lambda)$  itself can be considered as justifying alternating application of Add-and-Remove to the summary similarity criterion. Each iteration of this starts at the within-cluster average value of  $\lambda$  and corresponding threshold  $\pi = \lambda/2$  and finishes with finding a suboptimal  $S$ . The computation stops at the average  $\lambda$  coinciding with its previous value. In the beginning,  $\lambda$  can be taken as the average similarity over  $I$  if no expert-driven hypothetical value is available.

### 4.3 *Partitional, Additive and Injunctive Clusters: Iterative Extraction*

A similar approximal model can be considered as underlying a set of (not necessarily disjoint) similarity clusters  $S_1, S_2, \dots, S_K$ . Specifically, let a similarity cluster be specified by its binary belongingness vector  $s = (s_i)$  and a positive real  $\lambda$ , the cluster intensity, to make the binary scale compatible with that of similarity  $A$ . Then the model is defined by equations

$$a_{ij} = \biguplus_{k=1}^K \lambda_k s_i^k s_j^k + e_{ij}, \quad \text{for } i, j \in I, \quad (15)$$

where  $s^k = (s_i^k)$  and  $\lambda_k$  are  $k$ -th cluster belongingness vector and its intensity. The symbol  $\biguplus$  denotes an operation of integration of the binary values together with their intensities. We consider three versions of the operation: (a) additive clusters:  $\biguplus$  is just summation; (b) partitional clusters:  $\biguplus$  denotes the fact that clusters are disjoint, no overlapping; (c) injunctive clusters:  $\biguplus$  is maximum over  $k = 1, 2, \dots, K$ , that is, operation of inclusive disjunction.

The goal is to minimize the residuals  $e_{ij}$  with respect to the unknown relations  $R^k$  and intensities  $\lambda_k$ .

Additive cluster model was introduced, in the English language literature, by Shepard and Arabie in [26], and independently in a more general form embracing other cluster structures as well, by the author in mid-seventies in Russian ([12], see references in [13]). Injunctive clusters have not been considered in the literature, to our knowledge.

We maintain that cluster structures frequently are similar to that of the Solar system so that clusters hidden in data much differ with respect to their “contributions”. Then the iterative extraction method [12, 16] can be applied to find clusters one by one. Depending on the setting, that is, meaning of  $\biguplus$  in (15), one may use the following options:

(1) *Additive clusters*. The iterative extraction works as this:

- a. Initialization. Given a preprocessed similarity matrix  $A$ , compute the data scatter  $T = (A, A)$ . Put  $k = 0$ .
- b. General step. Add 1 to  $k$ . Find cluster  $S$  (locally) maximizing criterion  $g(S)$  in (14). Output that as  $S_k$ , the intensity of this cluster, the within-cluster average  $a(S)$  as  $\lambda_k$ , and its contribution to the data scatter,  $w_k = a(S)^2 |S|^2$ .
- c. Test. Check a stopping condition (see below). If it does hold, assign  $K = k$  and halt. Otherwise, compute the residual similarity matrix as  $A - \lambda_k s_k s_k^T$  and go back to General step with the residual matrix as  $A$ .

The stopping condition can be either reaching a prespecified number of clusters or contribution of the individual cluster has become too small or the total contribution of the so far found clusters has become too large. The individual cluster contributions are additive in this process. Moreover, the residual matrix in this process tends to 0 when  $k$  increases [14].

**Table 3** Additive clusters found at the Eurovision song contest dataset

No.	Cluster	Intensity	Contribution, %
1	Azerbaijan, Bulgaria, Greece, Russia, Serbia, Ukraine	70.0	21.43
2	Azerbaijan, Israel, Romania, Russia, Ukraine	49.5	7.13
3	Bulgaria, Greece, Italy, Romania, Spain	46.8	6.38
4	Azerbaijan, Poland, Ukraine	66.8	3.90
5	Italy, Portugal, Romania	53.0	2.46
6	Greece, Romania, Serbia	43.7	1.67

(2) *Partitional clusters*. This method works almost like the iterative extraction at the additive clustering model, except that here no residual matrix is considered, but rather the found clusters are removed from the set of entities.

- a. Initialization. Given a preprocessed similarity matrix  $A$ , compute the data scatter  $T = (A, A)$ . Put  $k = 0$ .
- b. General step. Add 1 to  $k$ . Find cluster  $S$  (locally) maximizing criterion  $g(S)$  in (14). Output that as  $S_k$ , the intensity of this cluster, the within-cluster average  $a(S)$  as  $\lambda_k$ , and its contribution to the data scatter,  $w_k = a(S)^2|S|^2$ .
- c. Test. Check a stopping condition (see below). If it does hold, assign  $K = k$  and halt. Otherwise, compute the residual entity set  $I = I - S_k$  and remove  $S_k$  from the similarity matrix. Go back to General step.

The stopping condition can be either reaching equation  $I = S_k$  or the situation at which the matrix  $A$  has no positive entries on the set of yet unclustered entities. The individual cluster contributions are additive in this process.

- (3) *Injunctive clusters*. Make a loop over  $i \in I$ . Run semi-average criterion Add-and-Remove( $S$ ) algorithm at  $S = \{i\}$  for each  $i$ . Remove those of the found clusters that overlap with others too much. This can be done by applying the same algorithm to the cluster-to-cluster similarity matrix; entries in this matrix are defined as proportional to the overlap values. An individual cluster over this matrix contains those clusters that overlap too much—only one of them should be left.

For an example, let us apply each of these three strategies to the Eurovision matrix, preliminarily made symmetric with zeroed diagonal entries.

- a. Additive clusters one by one: With the condition to stop when the contribution of an individual cluster becomes less than 1.5 % of the total data scatter, the algorithm found, in addition to the universal cluster  $I$  with the intensity equal to the similarity average, six more clusters (see Table 3). We can see that, say, pair Azerbaijan and Ukraine belong to three of the clusters and contribute, therefore, the summary intensity value  $70.0 + 49.5 + 66.8 = 186.3$  as the “model” similarity between them. This is, by the way, is greater than the observed similarity between them, 165.3.

**Table 4** Partitional clusters found one-by-one at the Eurovision song contest dataset

No.	Cluster	Intensity	Contribution, %
1	Azerbaijan, Bulgaria, Greece, Russia, Serbia, Ukraine	70.0	21.43
2	Italy, Portugal, Romania, Spain	56.1	5.50
3	Belgium, Netherlands	57.3	0.96
4	Germany, UK	45.3	0.60
5	France, Israel, Switzerland	11.6	0.12
6	Estonia, Poland	3.3	0.00

**Table 5** All four different injunctive clusters found at the Eurovision song contest dataset starting from every entity

No.	Cluster	Intensity	Contribution, %
1	Azerbaijan, Bulgaria, Greece, Russia, Serbia, Ukraine	70.0	21.43
2	Belgium, Netherlands	57.3	0.96
3	Bulgaria, Greece, Serbia	110.6	10.7
4	Italy, Portugal, Romania, Spain	56.1	5.50

- b. Partitional clusters one by one. Here the algorithm is run on the entities remaining unclustered after the previous step (see Table 4).

Here, in fact, there are only two meaningful clusters, East European and Latin South European; the other four contribute so little that should not be considered at all as possibly being “noise-generated”. The first of the clusters is just a replica of that in the additive clustering computation. Yet the second cluster is a different phenomenon, a combination of clusters 3 and 5 in the additive clusters results Table 3 cleaned of the Balkans.

- c. Injunctive clusters from every entity. The semi-average Add-and-Remove( $S$ ) algorithm has been applied starting from  $S = \{i\}$  for every  $i \in I$ . Most of the final clusters coincide with each other, so that there are very few different clusters. A surprising feature of the computation is that the algorithm sometimes moved out of the starting entity to finish with a cluster from which the entity was absent. Table 5 presents different clusters only.

According to the data recovery model, these clusters lead to a recovered similarity matrix as follows: first of all, the subtracted average value, 35.72, should be put at every entry. Then the Belgium/Netherlands link is to be increased by the intensity of cluster 2, 57.3. Similarly, the intensities of clusters 1 and 4 are to be added for any pair of entities within each. Then entries for pairs from cluster 3 are to be changed for  $35.7 + 110.6 = 146.3$ .

It should be reminded that of the four clusters in Table 5, only one is globally optimal, the first cluster, since the criterion is co-monotone with the contribution to data scatter, which is maximum at cluster one. Therefore, the other clusters have been found only because of the local nature of Add-and-Remove algorithm.

This is an example at which the local nature of the algorithm is of an advantage rather than a drawback. Because of the local nature, one can explore the local clustering structure. One can see that in this case, the structure is rather sharp, no intermediate clusters at all! Many entities remain out of the cluster structure. One of the clusters is of a special interest, cluster 3 consisting of Bulgaria, Greece and Serbia—hard-core Balkan countries. This comes on top of a less intensive cluster 1, containing all of the cluster 3. One can notice that this structure reflects more than 10 years of voting, and thus probably has nothing to do with the “quality” of songs presented to the contest. If so, the clusters reflect cultural interrelations rather than anything else.

On a more personal note, I can recall that a few years ago the media in the United Kingdom made an issue of the humble history of failure of the UK songs in the contest and decided to go for a win. They attracted one of the best composers, best performers and best relation managers—all this to arrive at one more disappointing failure. Looking at the clusters above, one cannot help but joining the scores of pessimists: the UK has no chances at the Eurovision song contest, unless the rules are changed to accommodate the cluster structure.

## 5 Applications

### 5.1 *Semantics of Domain-Specific Nouns*

The idea that semantics of domain-specific nouns lies in their relation to specific situations, functions, etc., a few decades back was not that obvious in cognitive sciences as it is now. In the absence of Internet, the researchers used the so-called sorting experiments to shed light on semantics of domain specific nouns [5, 23]. In a sorting experiment, a set of domain-specific words is specified and written down, each on a small card; a respondent is asked then to partition cards into any number of groups according to their perceived similarity among the nouns. Then a similarity matrix between the words can be drawn so that the similarity score between two words is defined as the number of respondents who put them together in the same cluster. A cognitive scientist may think that behind the similarity matrix can be some “additive” elementary meanings. Say, for a group of kitchenware terms, such as, say, “glass”, “cup”, “casserole” and “saucepan”, there is a common elementary meaning that they all are water reservoirs plus a meaning—cookware relating the two latter items. Depending on the social and cultural background, word clusters expressing the same meaning may have this or that intensity weight. The weights of different meanings relating two terms, are added together in the mind when considering them as a whole: this sounds a plausible hypothesis. Therefore, the additive cluster model can be adequate for the analysis of sorting experiment results.

In the analysis of similarities between 72 kitchenware terms, the iterative one-by-one extraction with Semi-average similarity Add-and-Remove algorithm found that

none of the clusters reflected logical or structural similarities between the kitchenware items; all the clusters related to the usage only. Specifically, three types of term communality were manifested in the clusters: (i) a cooking process, such as frying or boiling; (ii) a common consumption use, such as drinking or eating, and (iii) a common situation such as a banquet [5].

Let us show in a greater detail, how the individual clusters can be found at the matrix of similarity scores between elementary functions in Table 1. The average of non-diagonal entries in it is  $\bar{a} = 2.56$ . After subtracting this from the data the resulting similarity matrix upper triangle is:

	2	3	4	5	6	7	8	9
1	3.44	-1.56	-1.56	-0.56	0.44	-0.56	-0.56	-0.56
2		-1.56	-1.56	-0.56	-0.56	1.44	1.44	0.44
3			3.44	-1.56	-1.56	-1.56	-1.56	-1.56
4				-1.56	-1.56	-1.56	-1.56	-1.56
5					3.44	2.44	2.44	2.44
6						1.44	0.44	-0.56
7							2.44	0.44
8								-0.56

Since we are going to apply the iterative extraction method, the subtraction of 2.56 from the original data can be considered as transition to the residual matrix after the universal cluster, consisting of all the entities, is extracted. The contribution of the universal cluster to the original similarity data scatter is  $207.8/676 = 30.4\%$ .

Let us find an individual cluster, starting from  $S = \{5\}$ . Entity 6 has the maximum similarity with  $S$ ,  $a_{56} = 3.44$ . After it is added,  $S = \{5, 6\}$ . Let us find an entity that has the largest positive summary similarity to  $S$ . Entities 1, 2, 3 and 4 all have negative summary similarity to  $S$ . The summary similarity to  $S$  is equal to  $2.44 + 1.44 = 3.88$  for entity 7,  $2.44 + 0.44 = 2.88$  for 8, and  $2.44 - 0.56 = 1.88$  for 9. This leads to entity 7 being the candidate to be added to  $S$ . The average similarity from 7 to  $S$ ,  $3.88/2 = 1.94$  is greater than half the within- $S$  similarity,  $3.44/2 = 1.72$ , so that 7 joins  $S$ . Now  $S = \{5, 6, 7\}$  and the positive summary similarities with  $S$  are  $2.44 + 0.44 + 2.44 = 5.32$ , for 8, and  $2.44 - 0.56 + 0.44 = 2.32$ , for 9. The candidate 8 has its average similarity  $a(8, S) = 5.32/3 = 1.77$ . The average within- $S$  similarity is  $(3.44 + 2.44 + 1.44)/3 = 7.32/3 = 2.44$ , so that its half,  $2.44/2 = 1.22$ , is less than 1.77. Therefore, 8 joins in  $S$ . The remaining entity 9 summary similarity to  $S = \{5, 6, 7, 8\}$  is  $2.44 - 0.56 + 0.44 - 0.56 = 1.76$ . The average similarity  $a(9, S) = 1.76/4 = 0.44$  is less than the half of the within- $S$  average similarity,  $a(S) = 2.11$ , so that 9 cannot join  $S$ . Moreover, no entity should be removed from  $S$ , because each its element has a positive attraction to  $S$ . This leads to halting the process and outputting  $S = \{5, 6, 7, 8\}$ , together with its intensity  $a(S) = 2.11$  and the contribution  $w(S) = a(S)^2 \cdot 16/676 = 10.55\%$ . It is not difficult to see that this  $S$  is the best among all the clusters found at different  $j$ .

Table 6 presents 6 additive clusters found with the one-by-one process of additive clustering. The total contribution to data scatter is about 60 %. It should be noted that optimally adjusting weights of the six clusters with the orthogonal projection

**Table 6** Additive clusters found at Elementary functions dataset

No.	Cluster	Intensity	Contribution, %	Interpretation
1	1–9	2.56	30.44	Functions
2	5, 6, 7, 8	2.11	10.55	Power: growing
3	1, 2	3.44	7.02	Natural
4	3, 4	3.44	7.02	Power: decreasing
5	5, 9	2.44	3.54	Growing even
6	2, 7, 8	1.07	1.53	Slow growing

of matrix  $A - \bar{a}$  over binary matrices of the clusters would drastically increase that, up to about 94 % of the original data scatter. But in that case the additive property of individual contributions would be lost.

Unfortunately, the contribution weights do not lead to any sound advice for halting the process. Here, the process stops at six clusters as having reached the contribution of an individual cluster of 1.5 %. It is nice that all the found clusters have more or less reasonable, and in fact exclusive, interpretations.

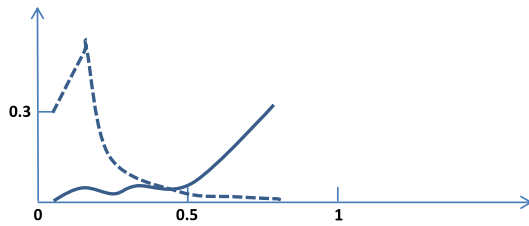
The same process, under the constraint that the next clusters are not to overlap those previously found, would lead to just three clusters in Table 6, Nos. 2, 3, and 4, leaving function 9 a singleton.

5.2 Determining Similarity Threshold by Combining Knowledge

In Mirkin et al. (2010) [17] partitional clusters of protein families in herpes viruses are found. The similarity between them is derived from alignments of protein amino acid sequences. Yet in viruses, the amino acid contents is highly changeable even in related genomes. This is why the similarity in [17] is measured over their neighborhoods [8, 29], that are subsets of proteins. The similarity between sets  $S$  and  $T$  is measured by the ratios  $|S \cap T|/|S|$  and  $|S \cap T|/|T|$  as well as by their average  $mbi(S, T) = (|S \cap T|/|S| + |S \cap T|/|T|)/2$ , which goes in line with the symmetric reformulation of the raw similarity data in Statement 2. This index spans interval from 0, no overlap, to 1, full coincidence.

At different similarity shifts, different numbers of clusters can be obtained, from 99 non-singleton clusters (of 740 entities) at the zero similarity shift to only 29 non-singleton clusters at the shift equal to 0.97 [17]. To choose a proper value of the shift, external information can be used—of functional activities of the proteins under consideration in [17]. Although function of most proteins under consideration was unknown, the set of pairs of functionally annotated proteins can be used to shed light onto potentially admissible values of the similarity shift. In each pair, the proteins can be synonymous (sharing the same function) or not. Because of a high simplicity of virus genomes, the synonymous proteins should belong in the same aggregate protein family, whereas proteins of different functions should belong in different protein families. The similarity shift value should be taken as that

**Fig. 2** Empirical percentage frequency functions (y-values) for the sets of synonymous pairs (solid line) and non-synonymous pairs (dashed line). The x-values represent the *mbc* similarity



between the sets of similarity values for synonymous and non-synonymous proteins. Then, after subtraction of this value, similarities between not synonymous HPFs get negative while those between synonymous HPFs remain positive. In [17], no non-synonymous pair has a greater *mbc* similarity than 0.66, which should imply that the shift value 0.67 confers specificity for the production of aggregate protein families. Unfortunately, the situation is less clear cut for synonymous proteins: although the similarities between them indeed are somewhat higher, 24 % pairs is less than 0.67. To choose a similarity shift that minimizes the error in assigning negative and positive similarity values, one needs to compare the distribution of similarity values in the set of synonymous pairs with that in the set of non-synonymous pairs (Fig. 2) and derive the intersection point similarity value as 0.42.

Thus the external knowledge of functional synonymy/non-synonymy reduces the set of candidate similarity shift values to two:

- (a)  $\lambda_0 = 0.67$  to guarantee specificity in that non-synonymous proteins are not be clustered together, and
- (b)  $\lambda_0 = 0.42$  to ensure the minimum misclassification error rate.

The final choice  $\lambda_0 = 0.42$  has been made over yet another type of external information: compatibility between the evolutionary histories of protein families derived in the framework of the maximum parsimony principle [17, 18] and the structure of gene arrangement in the genomes.

### 5.3 Consensus Clustering

Consensus clustering is an activity of summarizing a set of clusterings into a single clustering. This has become popular recently because after applying different clustering algorithms, or the same algorithm at different parameter settings, on a data set, one gets a number of different solutions. Consensus clustering seeks a unified cluster structure behind the solutions found (see, for example, [16, 30–32]). This author has contributed to the problem a few dozen decades ago, in Russian. Here some results of applying an approach from Mirkin and Muchnik [19] in the current setting will be reported. Some supplementary aspects of the least-squares consensus clustering are described in [16].

Consider a partition  $S = \{S_1, \dots, S_K\}$  on  $I$  and corresponding binary membership  $N \times K$  matrix  $Z = (z_{ik})$  where  $z_{ik} = 1$  if  $i \in S_k$  and  $z_{ik} = 0$ , otherwise



( $i = 1, \dots, N$ ,  $k = 1, \dots, K$ ). Obviously,  $Z^T Z$  is a diagonal  $K \times K$  matrix in which  $(k, k)$ -th entry is equal to the cardinality of  $S_k$ ,  $N_k = |S_k|$ . On the other hand,  $ZZ^T = (s_{ij})$  is a binary  $N \times N$  matrix in which  $s_{ij} = 1$  if  $i$  and  $j$  belong to the same class of  $S$ , and  $s_{ij} = 0$ , otherwise. Therefore,  $(Z^T Z)^{-1}$  is a diagonal matrix of the reciprocals  $1/N_k$  and  $P_Z = Z(Z^T Z)^{-1}Z^T = (p_{ij})$  is an  $N \times N$  matrix in which  $p_{ij} = 1/N_k$  if both  $i$  and  $j$  belong to the same class  $S_k$ , and  $p_{ij} = 0$ , otherwise. Matrix  $P_Z$  represents the operation of orthogonal projection of any  $N$ -dimensional vector  $x$  onto the linear subspace  $L(Z)$  spanning the columns of matrix  $Z$ .

A set of partitions  $R^u$ ,  $u = 1, 2, \dots, U$ , along with the corresponding binary membership  $N \times L_u$  matrices  $X^u$ , found with various clustering procedures, can be thought of as proxies for a hidden partition  $S$ , along with its binary membership matrix  $Z$ . Each of the partitions can be considered as related to the hidden partition  $S$  by equations

$$x_{il}^u = \sum_{k=1}^K c_{kl}^u z_{ik} + e_{ik}^u \quad (16)$$

where coefficients  $c_{kl}^u$  and matrix  $z_{ik}$  are to be chosen to minimize the residuals  $e_{ik}^u$ .

By accepting the sum of squared errors  $E^2 = \sum_{i,k,u} (e_{ik}^u)^2$  as the criterion to minimize, one immediately arrives at the optimal coefficients being orthogonal projections of the columns of matrices  $X^u$  onto the linear subspace spanning the hidden matrix  $Z$ . More precisely, at a given  $Z$ , the optimal  $K \times L_u$  matrices  $C^u = (c_{kl}^u)$  are determined by equations  $C^u = Z(Z^T Z)^{-1}X^u$ . By substituting these in equations (16), the square error criterion can be reformulated as:

$$E^2 = \sum_{u=1}^U \|X^u - P_Z X^u\|^2 \quad (17)$$

where  $\|\cdot\|^2$  denotes the sum of squares of the matrix elements.

The problem is to find a partition with the membership matrix  $Z$  minimizing criterion (17). This criterion seems rather original, never mentioned in the current literature, to this author's knowledge. It is not difficult to show that the criterion can be reformulated in terms of the so-called consensus similarity matrix. To this end, let us form  $N \times L$  matrix  $X = (X^1 \ X^2 \ \dots \ X^U)$  where  $L = \sum_{u=1}^U L_u$ . The columns of this matrix correspond to clusters  $R_l$  that are present in partitions  $R^1, \dots, R^U$ . Then the least squares criterion can be expressed as  $E^2 = \|X - P_Z X\|^2$ , or equivalently, as  $E^2 = \text{Tr}((X - P_Z X)(X - P_Z X)^T)$  where  $\text{Tr}$  denotes the trace of  $N \times N$  matrix, that is, the sum of its diagonal elements, and  $^T$ , the transpose. By opening the parentheses in the latter expression, one can derive that  $E^2 = \text{Tr}(XX^T - P_Z XX^T)$ . Let us denote  $A = XX^T$  and take a look at  $(i, j)$ -th element of this matrix  $a_{ij} = \sum_l x_{il}x_{jl}$  where summation goes over all clusters  $R_l$  of all partitions  $R^1, R^2, \dots, R^U$ . Obviously,  $a_{ij}$  equals the number of those partitions  $R^1, R^2, \dots, R^U$  at which  $i$  and  $j$  are in the same class. This matrix is referred to in the literature as the consensus matrix.

**Table 7** Consensus clustering methods involved in the experiments

No.	Method	Author(s)	Reference
1	Bayes	Wang et al.	[32]
2	Vote	Dimitriadi et al.	[4]
3	CVote	Ayad, Kamel	[1]
4	Borda	Sevillano et al.	[25]
5	Fusion	Guenoche	[7]
6	CSPA	Strehl, Ghosh	[30]
7	MCLA	Strehl, Ghosh	[30]

The latter expression can be reformulated thus as

$$E^2 = NU - \sum_{k=1}^K \sum_{i,j \in S_k} a_{ij} / N_k$$

This leads us to the following statement.

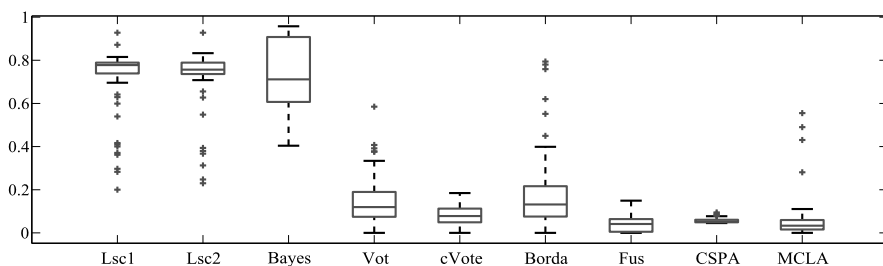
**Statement 7** *A partition  $S = \{S_1, \dots, S_K\}$  is an ensemble consensus clustering if and only if it maximizes criterion*

$$g(S) = \sum_{k=1}^K \sum_{i,j \in S_k} a_{ij} / N_k \quad (18)$$

where  $A = (a_{ij})$  is the consensus matrix between entities for the given set of partitions.

Criterion (18) is but the sum of the semi-average criteria for the clusters  $S_1, \dots, S_K$ . Therefore, the iterative extraction algorithm in its partitional clusters format is applicable here. We compared the performances of this algorithm and a number of up-to-date algorithms of consensus clustering (see Table 7) [27].

These algorithms have been compared with two versions of the iterative extraction partitional clusters method above differing by the condition whether the option of zeroing all the diagonal entries of the similarity matrix has been utilized or not (Lsc1 and Lsc2). These build a cluster by applying Add-and-Remove( $i$ ) for every  $i$  available and choosing that of the clusters found with the maximum contribution. Three types of datasets have been used: (a) datasets from the Irvine Data Repository, (b) generated synthetic datasets, and (c) specially drawn artificial 2D shapes. The results are more or less similar to each other. Here we present only results of applying the algorithms to the Wisconsin Diagnostic Breast Cancer (WDBC) dataset from UCI Data Repository (569 entities, 30 features, two classes). The setting was as follows. The number of clusters is considered unknown. Therefore, first, the number of clusters  $k$  is generated randomly in the interval from 2 to 15 inclusive. Then  $k$ -means runs using a random initialization. The resulting partition is stored. This is



**Fig. 3** Comparison of the accuracy of consensus clustering algorithms at WDBC dataset

repeated 20 times, after which each of the nine algorithms applies to find a corresponding consensus partition. This partition is compared with that 2-class partition of WDBC dataset that is supplied with it, one class is of malignant cases, the other, benign. The scoring function reported is the accuracy, a measure of the quality of prediction. Given a consensus partition, overlaps of each of its clusters with the benign and malignant classes are computed, and that the larger of the two is taken as the “prediction”. Then the union of all the “predictions” is taken and its proportion in the entire dataset is the accuracy. Figure 3 presents the box-plot of all the accuracy values after a hundred of repetitions of the above.

This shows that Semi-average Criterion Add-and-Remove algorithm, in spite of its ‘greediness’, can be successfully applied in some specific situations.

## 6 Conclusion

The chapter describes a method for finding individual similarity clusters, that can be presented in several perspectives—summary similarity criterion, semi-average criterion, spectral clustering criterion and approximation criterion. The clustering criterion involves, in different forms, the concept of similarity threshold, or similarity shift—a value subtracted from all the similarity matrix entries. The summary similarity criterion involves a constant similarity threshold, whereas the semi-average criterion can be thought of as a summary criterion with a changeable similarity threshold that keeps to the half mean of the within cluster similarities. The threshold can be used for bridging different aspects of the phenomenon under study together. This is demonstrated in Sect. 5.2, in which the final choice of clustering involves the protein function and gene arrangement in the genomic circle, in addition to the original similarity derived from protein sequences.

The criterion leads to nice properties of the clusters. First of all, the similarity data always can be made symmetric by just redistributing similarity symmetrically. Second, clusters are quite tight over average similarities of individual entities with them: these within clusters are always greater than those outside clusters; the half average within cluster similarity being a threshold physically separating the two in the case of the semi-average criterion. Third, the recommended locally optimizing

the criterion method, Add-and-Remove, is much intuitive: each step is based on choosing an entity according to its average similarity to the fragment of cluster that has been built already in the process. Fourth, unlike methods for finding global optima, this one leads to recovery of the local cluster structure of the data, probably a single most important innovation proposed in this chapter.

The Semi-average criterion, in fact, does not hang out as a sole object. It is closely related to the least-squares criterion utilized for clustering data in the conventional setting of entity-to feature data tables [16]. It also closely follows the concept of consensus clustering. An original method for fuzzy clustering by exploiting the same principles of one-by-one extraction of clusters, has shown rather good results in experiments [20].

**Acknowledgements** In the end, I would like to express my gratitude for the partial support of this work to the International Laboratory of Decision Choice and Analysis at NRU HSE (headed by F. Aleskerov) and the Laboratory of Algorithms and Technologies for Network Analysis NRU HSE Nizhny Novgorod by means of RF government grant ag. 11.G34.31.0057 (headed by V. Kalyagin).

## References

1. Ayad, H., Kamel, M.: On voting-based consensus of cluster ensembles. *Pattern Recognit.* **43**, 1943–1953 (2010)
2. Bader, G.D., Hogue, C.W.V.: An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinform.* **4**, 2 (2003)
3. Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering gene expression patterns. *J. Comput. Biol.* **6**, 281–297 (1999)
4. Dimitriadou, E., Weingessel, A., Hornik, K.: A combination scheme for fuzzy clustering. In: *Proceedings of the 2002 AFSS International Conference on Fuzzy Systems*, pp. 332–338 (2002)
5. Frumkina, R., Mirkin, B.: Semantics of domain-specific nouns: a psycho-linguistic approach. *Not. Russ. Acad. Sci. Lang. Lit.* **45**(1), 12–22 (1986) (in Russian)
6. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.* **18**, 30–55 (1989)
7. Guenoche, A.: Consensus of partitions: a constructive approach. *Adv. Data Anal. Classif.* **5**, 215–229 (2011)
8. Jarvis, R.A., Patrick, E.A.: Clustering using a similarity measure based on shared nearest neighbors. *IEEE Trans. Comput.* **22**, 1025–1034 (1973)
9. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.* **49**(2), 291–307 (1970)
10. Kupershtoh, V., Mirkin, B.: A problem for automatic classification. In: Bagrinowski, K. (ed.) *Mathematical Methods for Economics*, pp. 39–49. Siberian Branch of Nauka Publisher, Novosibirsk (1968) (in Russian)
11. Kupershtoh, V., Mirkin, B., Trofimov, V.: Sum of within partition similarities as a clustering criterion. *Autom. Remote Control* **37**(2), 548–553 (1976)
12. Mirkin, B.: *Analysis of Categorical Features*. Finansy i Statistika, Moscow (1976). 166 pp. (in Russian)
13. Mirkin, B.: Additive clustering and qualitative factor analysis methods for similarity matrices. *J. Classif.*, **4**, 7–31 (1987). Erratum **6**, 271–272 (1989)
14. Mirkin, B.: *Mathematical Classification and Clustering*. Kluwer Academic, Dordrecht (1996)
15. Mirkin, B.: *Core Concepts in Data Analysis: Summarization, Correlation, Visualization*. Springer, London (2011)

16. Mirkin, B.: *Clustering: A Data Recovery Approach*, 2nd edn. Chapman and Hall, Boca Raton (2012)
17. Mirkin, B.G., Camargo, R., Fenner, T., Loizou, G., Kellam, P.: Similarity clustering of proteins using substantive knowledge and reconstruction of evolutionary gene histories in herpesvirus. *Theor. Chem. Acc.* **125**(3–6), 569–581 (2010)
18. Mirkin, B., Fenner, T., Galperin, M., Koonin, E.: Algorithms for computing parsimonious evolutionary scenarios for genome evolution, the last universal common ancestor and dominance of horizontal gene transfer in the evolution of prokaryotes. *BMC Evol. Biol.* **3**, 2 (2003). [www.biomedcentral.com/1471-2148/3/2/](http://www.biomedcentral.com/1471-2148/3/2/)
19. Mirkin, B., Muchnik, I.: Geometric interpretation of clustering criteria. In: Mirkin, B. (ed.) *Methods for Analysis of Multidimensional Economics Data*, pp. 3–11. Nauka Publishers (Siberian Branch), Novosibirsk (1981) (in Russian)
20. Mirkin, B., Nascimento, S.: Additive spectral method for fuzzy cluster analysis of similarity data including community structure and affinity matrices. *Inf. Sci.* **183**, 16–34 (2012)
21. Newman, M.E.J.: Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **103**(23), 8577–8582 (2006)
22. Newman, M., Girvan, M.: Finding and evaluating community structure in networks. *Phys. Rev. E* **69**, 026113 (2004)
23. Rosenberg, S., Kim, M.P.: The method of sorting as a data-gathering procedure in multivariate research. *Multivar. Behav. Res.* **10**, 489–502 (1975)
24. Satarov, G.A.: A non-intrusive knowledge evaluation method. Personal communication (1981)
25. Sevillano Dominguez, X., Socoro Carrie, J.C., Alias Pujol, F.: Fuzzy clusters combination by positional voting for robust document clustering. *Procesamiento del Lenguaje Natural* **43**, 245–253 (2009)
26. Shepard, R.N., Arabie, P.: Additive clustering: representation of similarities as combinations of overlapping properties. *Psychol. Rev.* **86**, 87–123 (1979)
27. Shestakov, A., Mirkin, B.G.: Least square consensus clustering: criteria, methods, experiments. In: *Advances in Information Retrieval. LNCS*, vol. 7814, pp. 764–767 (2013)
28. Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 888–905 (2000)
29. Small, H.: Co-citation in the scientific literature: a new measure of the relationship between two documents. *J. Am. Soc. Inf. Sci.* **24**, 265–269 (1973)
30. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* **3**, 583–617 (2002)
31. Swift, S., Tucker, A., Vinciotti, V., Martin, N., Orengo, C., Liu, X., Kellam, P.: Consensus clustering and functional interpretation of gene expression data. *Genome Biol.* **5**, R94 (2004)
32. Wang, H., Shan, H., Banerjee, A.: Bayesian cluster ensembles. In: *Proceedings of the Ninth SIAM International Conference on Data Mining*, pp. 211–222 (2009)

# Kernel Principal Component Analysis: Applications, Implementation and Comparison

Daniel Olsson, Pando Georgiev, and Panos M. Pardalos

**Abstract** Kernel Principal Component Analysis (KPCA) is a dimension reduction method that is closely related to Principal Component Analysis (PCA). This report gives an overview of kernel PCA and presents an implementation of the method in MATLAB. The implemented method is tested in a transductive setting on two data bases. Two methods for labeling data points are considered, the nearest neighbor method and kernel regression, together with some possible improvements of the methods.

**Keywords** Reproducing kernel Hilbert space · Kernel principal component analysis · Dimension reduction · Nearest neighbor · Kernel regression

## 1 Introduction

Data mining is concerned with finding meaningful patterns in large sets of data. The swift development in computation power that has taken place in recent years has increased the amount of available data in an avalanche-like fashion (consider for example, the Human Genome Project, in which the whole human DNA is stored in a database). The task of analyzing these gigantic amounts of data can appear overwhelming. Naturally, there is a need for tools that can aid the analyst in identifying the interesting features of the data.

In the past decades, a group of methods known as dimension reduction methods have gained wide-spread attention. As the name implies, dimension reduction

---

D. Olsson  
Royal Institute of Technology (KTH), Stockholm, Sweden  
e-mail: [daolsson@kth.se](mailto:daolsson@kth.se)

P. Georgiev (✉) · P.M. Pardalos  
Center for Applied Optimization, University of Florida, Gainesville, USA  
e-mail: [pandogeorgiev@ufl.edu](mailto:pandogeorgiev@ufl.edu)

P.M. Pardalos  
e-mail: [pardalos@ufl.edu](mailto:pardalos@ufl.edu)

methods seek to solve this problem by reducing the number of dimensions of the data, while retaining most of the information in the data set. By removing redundant dimensions, it becomes easier to recognize patterns or tendencies in the remaining data. Some examples of dimension reduction methods are Support Vector Machines and Random Projections (e.g., [10] and [1, 3]).

In this paper, one particular dimension reduction method is considered, namely Kernel Principal Component Analysis (KPCA). The method is implemented in MATLAB and applied to data sets. In addition, different ways of modifying kernel PCA in order to improve the results on these particular data sets will be considered.

The outline of the report is the following. Section 4 provides the description of kernel PCA, which is an extension of standard PCA, and also consider other techniques that are used. Section 5 contains the numerical experiments that were performed on two databases. The results are commented along with a discussion in Sect. 6. Central parts of the MATLAB-source code that was used in order to perform kernel PCA can be found in the [Appendix](#).

## 2 Principal Component Analysis

Principal Component Analysis (PCA) is a method for reducing the number of dimensions of a data set, while retaining as much of the information in the data as possible. The data is projected onto a subspace, spanned by the eigenvectors that captures most of the variance of the data, while the remaining dimensions are neglected.

Let  $X$  be an  $n \times d$ -matrix, representing a set of  $n$  data points  $x_i \in \mathbb{R}^d$ ,  $i = 1, \dots, n$ , where each data point is a row vector. Then the  $d \times d$  covariance matrix  $C$  of the data set is given by

$$C = \frac{1}{n} (X - \bar{X})^T (X - \bar{X}), \quad (1)$$

where  $\bar{X}$  is an  $n \times d$ -matrix in which each row is the center of the data points. By assuming that the data in  $X$  is centered, the covariance matrix can be expressed as

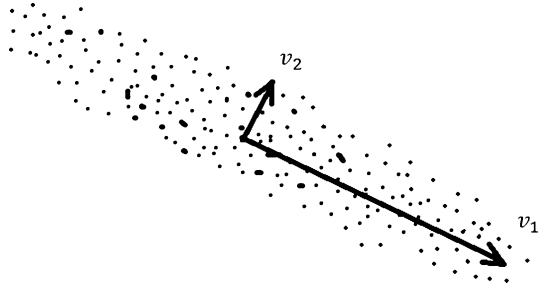
$$C = \frac{1}{n} X^T X. \quad (2)$$

The eigenvalues of  $C$  can then be obtained by solving the equation

$$\lambda V = CV, \quad (3)$$

where  $\lambda$  is an  $n \times n$  matrix containing the eigenvalues in the diagonal and  $V$  contains the corresponding (normalized) eigenvectors. The final step is to project the data of  $X$  onto the new  $l$ -dimensional subspace that is spanned by the  $l$  eigenvectors that

**Fig. 1** Principal Component Analysis



have the largest corresponding eigenvalues. The larger an eigenvalue is in relation to the other eigenvalues, the more of the variance in the data it captures. The projection gives a set of new points  $\{\chi_1, \dots, \chi_n\}$ , and is obtained by the following formula

$$\chi = C V^l, \quad (4)$$

where  $V^l$  is an  $n \times l$ -matrix of the  $l$  eigenvectors with largest eigenvalues as columns. The subspace spanned by the eigenvectors in  $V^l$  captures the largest possible amount of the total variance of the data in  $X$ . In order to capture more of the variance, more dimensions need to be included into the projection.

In Fig. 1, PCA is performed on a two-dimensional data set. As can be seen in the figure, a projection of the data onto the eigenvector  $v_1$  captures most of the variance of the data set.

A further description of PCA can be found in [6].

### 3 Reproducing Kernel Hilbert Spaces

Recall one of the possible constructions of Reproducing Kernel Hilbert Spaces (RKHS) (see [2, 5] for more details). Let  $\mathbf{X} \subset \mathbb{R}^m$  and  $K : \mathbf{X} \times \mathbf{X} \rightarrow \mathbb{R}$  be a continuous and symmetric function such that  $\{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^n$  is positive definite for every  $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbf{X}$  and every  $n$ , i.e.  $\sum_{i,j=1}^n c_i c_j K(x_i, x_j) \geq 0$  for any  $n \in \mathbb{N}$  and any choice of  $x_i \in X$  and  $c_i \in \mathbb{R}$  ( $i = 1, \dots, n$ ). Note  $K(x, x) \geq 0$  for all  $x$ . The mapping  $\mathbf{K}$  is called *positive definite kernel*.

Let  $H_0$  be the linear span of the functions  $\{K(\mathbf{x}, \cdot), \mathbf{x} \in \mathbf{X}\}$ :

$$H_0 = \left\{ f : \mathbf{X} \rightarrow \mathbb{R} : f(\cdot) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \cdot), \mathbf{x}_i \in \mathbf{X}, \alpha_i \in \mathbb{R}, n \in \mathbb{N} \right\}.$$



Define an inner product in  $H_0$  by

$$\langle f, g \rangle = \sum_{i=1}^n \sum_{j=1}^{n'} \alpha_i \beta_j K(\mathbf{x}_i, \mathbf{x}'_j) \quad (5)$$

where  $f, g \in H_0$ ,  $f(\cdot) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \cdot)$ ,  $g(\cdot) = \sum_{i=1}^{n'} \beta_i K(\mathbf{x}'_i, \cdot)$ .

The *reproducing property*

$$f(\mathbf{x}) = \langle f, K(\mathbf{x}, \cdot) \rangle \quad \forall \mathbf{x} \in \mathbf{X}$$

follows from (5).

The induced norm  $\|\cdot\|_K$  in  $H_0$  is  $\|f\|_K = \sqrt{\langle f, f \rangle}$ . It is indeed a norm, since if  $\|f\| = 0$ , then  $f(\mathbf{x}) = \langle f, K(\mathbf{x}, \cdot) \rangle \leq \|f\| \|K(\mathbf{x}, \cdot)\| = 0 \quad \forall \mathbf{x}$ .

The completion of  $(H_0, \|\cdot\|_K)$  is called *Reproducing Kernel Hilbert Space*.

The so called *kernel trick*

$$K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle \quad \text{for every } \mathbf{x}, \mathbf{y} \in \mathbf{X}$$

where  $\Phi(\mathbf{x}) = K(\mathbf{x}, \cdot)$  is called *feature map*, follows again from (5).

We give two typical examples of positive definite kernels. The first is the Gaussian kernel  $K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$  defined by  $K(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/c^2)$  ( $c > 0$ ).

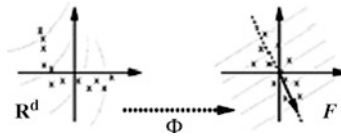
The second is the polynomial kernel of degree  $p$ ,  $K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$  defined by  $K_p(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^p$ . In the case when for  $m = 2$ ,  $p = 3$ , we have (for  $\mathbf{x} = (x, y)$ ,  $\mathbf{x}' = (x', y')$ ),

$$\begin{aligned} \Phi(\mathbf{x}) &= (x^3, y^3, \sqrt{3}x^2y, \sqrt{3}xy^2, \sqrt{3}x^2, \sqrt{3}y^2, \sqrt{3}xy, \sqrt{3}x, \sqrt{3}y, 1) \\ \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle &= K((x, y), (x', y')). \end{aligned}$$

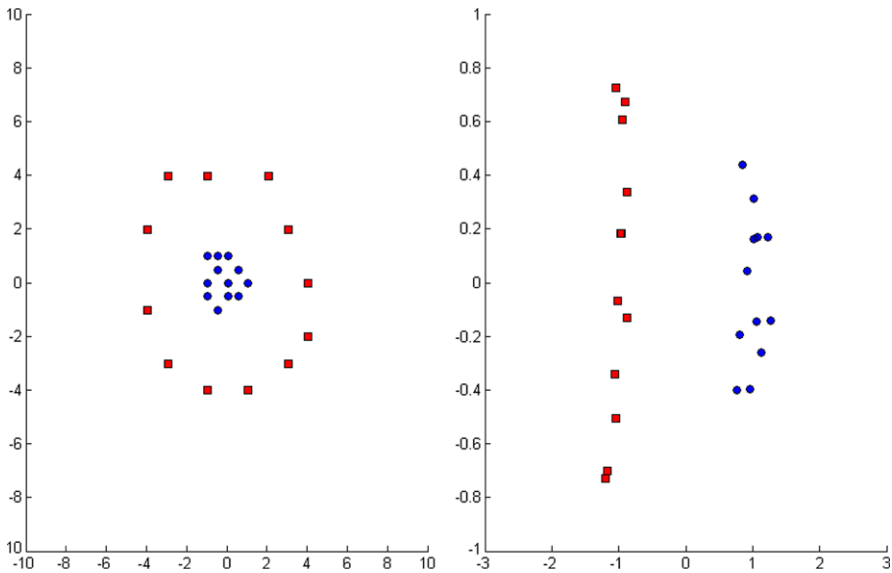
## 4 Kernel PCA

Kernel PCA uses the same basic idea as PCA, namely that it seeks to project the set of data onto a low-dimensional subspace that captures the highest possible amount of variance in the data. However, while PCA performs a linear separation of the data in the original space  $R^d$ , kernel PCA embeds the data into a high dimensional space  $F$ , RKHS—called the feature space, and performs a linear separation in that space instead. The advantage with this is that in  $R^d$ , it allows for nonlinear features to be extracted from the data.

Thus, while kernel PCA looks for linear features in RKHS (feature space), these features correspond to nonlinear features in the original lower dimensional space  $R^d$ , as shown in Fig. 2. The data in the feature space is then projected onto a low-dimensional subspace, spanned by the eigenvectors that capture most of the variance.



**Fig. 2** Data points in original space and feature space (von Luxburg 2006). Although a cluster of points may be scattered in a nonlinear pattern in the original space (*left*), they might form a pattern that is linear in the high dimensional feature space (*right*)



**Fig. 3** An illustration of kernel PCA. In the *left plot*, the two circles of points (*red* and *blue*) cannot be separated linearly from each other (for example by linear PCA). Kernel PCA embeds the points into a high dimensional space and then projects them onto a subspace spanned by principal eigenvectors, resulting in the *right plot*. The two classes of points in the *right plot* can be linearly separated from each other (Color figure online)

The advantage of kernel PCA over standard (linear) PCA is illustrated in Fig. 3. In the original space (left), the two types of data points cannot be linearly separated. However, by extracting nonlinear features, the points can be sorted according to their class (right).

For a further description of kernel PCA, see [12] and [13, Chap. 6].

## 4.1 Mathematical Formulation

Let the set of  $n$  data points be represented by

$$X = \{x_1, \dots, x_n\}, \quad (6)$$

where  $x_i \in R^d$ ,  $i = 1, \dots, n$ . It is assumed that the data in  $X$  is centered in the original space  $R^d$ , so that

$$\sum_{i=1}^n x_i = 0. \quad (7)$$

This condition can be met by using the formula

$$\tilde{x}_k = x_k - \frac{1}{n} \sum_{i=1}^n x_i, \quad (8)$$

where  $\tilde{x}_k$  is the  $k$ th data point in the new, centered set.

The following is a description of how kernel PCA is performed on a data set  $X$ , consisting of  $n$  data points.

Let  $\Phi$  be a mapping from the original space  $R^d$  of the data vectors, into a dot product space  $F$ , so that

$$\Phi : R^d \rightarrow F, \quad (9)$$

where  $F$  is a RKHS, referred to as the feature space. Thus,  $\Phi(x_i)$  represents the image of the data vector  $x_i$  in feature space.

Let the kernel matrix  $K$  be a symmetric and positive semidefinite  $n \times n$ -matrix, with its elements defined by the inner product of all pairs of points  $\Phi(x_i)$  and  $\Phi(x_j)$  in feature space so that

$$K_{ij} \triangleq (\Phi(x_i) \cdot \Phi(x_j)), \quad i, j = 1, \dots, n. \quad (10)$$

It is not necessary to know the values of  $\Phi(x_i)$  and  $\Phi(x_j)$  explicitly in order to compute the kernel matrix  $K$ . Instead, the elements of  $K$  can be calculated by computing the pairwise relation of all points  $x_i, x_j \in X$  in the original space  $R^d$ , through some kernel function  $k(x_i, x_j)$ , so that

$$K_{ij} = k(x_i, x_j), \quad i, j = 1, \dots, n. \quad (11)$$

The feature space  $F$  and the mapping  $\Phi$  are implicitly defined by the kernel function that is used. In this report, the Gaussian radial basis function (RBF) has been used (equation (17)).

The data points represented in the kernel matrix  $K$  are assumed to be centered in feature space, in the sense that

$$\sum_{i=1}^n \Phi(x_i) = 0. \quad (12)$$

For a non-centered set, this centering can be achieved by the inner product computation

$$(\tilde{K})_{ij} = ((\Phi(x_i) - \bar{\Phi}) \cdot (\Phi(x_j) - \bar{\Phi})) \quad (13)$$

where  $\tilde{K}$  is the centered kernel matrix and  $\overline{\Phi}$  is the center point in the feature space, given by

$$\overline{\Phi} = \frac{1}{n} \sum_{i=1}^n \Phi(x_i). \quad (14)$$

A derivation of the centering in feature space is found in the [Appendix](#).

The eigenvectors of the (centered) kernel matrix  $\tilde{K}$  can be obtained by solving the equation

$$\tilde{K}V = \lambda V, \quad (15)$$

where the columns of  $V$  represent the eigenvectors and  $\lambda$  is a matrix in which the diagonal consist of the corresponding eigenvalues and all other elements in  $\lambda$  are zero. The eigenvectors in  $V$  are assumed to be ordered according to the size of their corresponding eigenvalues, in descending order.

The representation of the low-dimensional points  $\{\chi_1, \dots, \chi_n\}$ , with  $\chi_i \in R^l$ ,  $i = 1, \dots, n$ , can be obtained by projecting the kernel matrix  $\tilde{K}$  onto the  $l$  eigenvectors that have the largest corresponding eigenvalues, by using the formula

$$\chi_i = \tilde{K}_i V^l, \quad (16)$$

where  $\tilde{K}_i$  is the  $i$ th row of  $\tilde{K}$ , and  $V^l$  is the matrix consisting of the  $l$  first eigenvector columns in  $V$ .

The kernel matrix is defined as (10) and thus contains the inner product of each pair of points  $x_i, x_j$  in feature space  $F$ . However, the value of  $K_{ij}$  is determined by some kernel function  $k(x_i, x_j)$ . This is a consequence of Mercer's Theorem, which states that any symmetric, positive semidefinite matrix can be regarded as a kernel matrix, that is as an inner product matrix in some space, which in the case of kernel PCA is the feature space.

**Theorem 1** (Mercer's Theorem) *If  $k$  is a continuous kernel of a positive integral operator, then it is possible to construct a mapping into a space where  $k$  acts as a dot product.*

The implication of Mercer's theorem for kernel PCA is that the inner product  $(\Phi(x_i) \cdot \Phi(x_j))$  can be substituted by the kernel function  $k(x_i, x_j)$  that we choose. There are many different kernel functions available, for example the Gaussian radial basis function<sup>1</sup>

$$K_{ij} = k(x_i, x_j) = \exp(-\sigma \|x_i - x_j\|^2). \quad (17)$$

As can be seen from (17),  $K_{ij}$  tends to go toward zero if the points  $x_i$  and  $x_j$  are far away from each other in  $R^d$ , while  $K_{ij}$  is close to one as the two points are near

---

<sup>1</sup>An alternative formulation of the radial basis function is  $k(x, y) = \exp(-\frac{\|x-y\|^2}{2\sigma^2})$ , used in for example [11, 12]. In this report, the definition in (17) is consistently used.

each other in  $R^d$ . As seen from the definition in (10),  $K_{ij}$  is the inner product of  $\Phi(x_i)$  and  $\Phi(x_j)$ . Thus, the following relationship between the points  $x_i, x_j$  in the original space  $R^d$  and their images  $\Phi(x_i), \Phi(x_j)$  in the feature space is valid:

If  $K_{ij} = 0$ , that is,  $x_i$  and  $x_j$  distant from each other in  $R^d$ , then  $\Phi(x_i)$  and  $\Phi(x_j)$  are orthogonal in feature space. Likewise, if  $K_{ij} = 1$ , that is,  $x_i$  and  $x_j$  are near each other in  $R^d$ , then  $\Phi(x_i)$  and  $\Phi(x_j)$  are parallel (and if  $\Phi(x_i)$  and  $\Phi(x_j)$  are assumed to be normal vectors in  $F$ , the two points are equal to each other).

The consequence of this is that data points from a high-dimensional data set  $X$  that are geometrically close to each in the space  $R^d$ , will be “almost” parallel in feature space. Furthermore, if the points in feature space are normal vectors in the sense that

$$\|\Phi(x_i)\| = 1, \quad i = 1, \dots, n, \quad (18)$$

then points that are almost parallel will be close to each other, as they are located on the hypersphere of radius one in the feature space.

## 4.2 The Transduction Problem

The transduction problem is the problem of completing the labeling of a partially labeled set. Many of the problems that are considered in this report have been posed in this form. The idea is to use the data and known labels of a training set in order to label the data in the test set. Naturally, a large training set tends to increase the classification performance, as there are more features that can be extracted. It is possible to compute a kernel matrix based on the test points relation to the training points, and then project it onto a low dimensional space. That is one way in which a test set consisting of points of different classes can potentially be separated, and thus labeled, according to class.

Let the first  $n_{\text{tr}}$  data points in  $X$  belong to the training set

$$X_{\text{tr}} = \{x_1, \dots, x_{n_{\text{tr}}}\} \subset X \quad (19)$$

and that the remaining  $n_{\text{tt}}$  of the data points in  $X$  belong to the test set

$$X_{\text{tt}} = \{x_{n_{\text{tr}}+1}, \dots, x_{n_{\text{tr}}+n_{\text{tt}}}\} \subset X, \quad (20)$$

so that  $n_{\text{tr}} + n_{\text{tt}} = n$ . Furthermore, let each data point  $x_i \in X$  belong to a class  $c_i \in R$  and assume that the class of each point in  $X_{\text{tr}}$  is known, and stored in the label set

$$y = \{y_1, \dots, y_{n_{\text{tr}}}\}. \quad (21)$$

Thus, the class of the  $i$ th data point in the training set is given by the element  $y_i \in R$  (in the case of binary classification,  $y_i \in \{+1, -1\}$ ).

The  $n_{\text{tr}} \times n_{\text{tt}}$ -test kernel matrix  $K_{\text{tt}}$  is formed by computing the relation between the points in the test set  $X_{\text{tt}}$  and the training set  $X_{\text{tr}}$  with the kernel function

$$K_{ij} = k(x_i, x_j) = \exp(-\sigma \|x_i - x_j\|^2), \quad (22)$$

where  $x_i \in X_{\text{tr}}$  and  $x_j \in X_{\text{tt}}$ , for  $i = 1, \dots, n_{\text{tr}}$  and  $j = 1, \dots, n_{\text{tt}}$ . The low dimensional projection of the test points

$$\chi_{\text{tt}} = \{\chi_1, \dots, \chi_{n_{\text{tt}}}\} \quad (23)$$

is obtained by projecting the test kernel matrix onto the  $l$  first eigenvectors that are extracted from the training kernel matrix. The test kernel matrix is assumed to be centered in the same sense as (12). However, as the test kernel matrix is not a square matrix, the formula for centering in feature space is slightly different (see the [Appendix](#) for details).

The resulting projection is given by

$$\chi_{\text{tt}} = K_{\text{tt}} V^l, \quad (24)$$

in which the projected test points  $\chi_i \in R^l$ , for  $i = 1, \dots, n_{\text{tt}}$  and  $V^l$  is the matrix consisting of the  $l$  first eigenvector columns in  $V$ .

### 4.3 The Nearest Neighbor Method

The nearest neighbor method can be used in transduction problems, i.e. in the classification of a partially labeled set [4]. The idea is to use reference points (with known labels) in addition to the test points, and to assign the same label to a test point as that of its geometrically nearest reference point. The reference points are randomly selected from the same data pool as the test set and are thus expected to be distributed evenly among the test points, if the size of the two sets is large. In the experiments described in this paper, the same points were used in the reference set as in the training set ( $X_{\text{ref}} = X_{\text{tr}}$ ).

Let  $X_{\text{ref}} \subset X$  be the set of  $n_{\text{ref}}$  reference points, where each point belongs to the class given by the corresponding element in the label vector  $y_{\text{ref}}$ . Furthermore, let  $K_{\text{ref}}$  be a  $n_{\text{ref}} \times n_{\text{tr}}$ -kernel matrix based on  $X_{\text{ref}}$  and  $X_{\text{tr}}$ , through the kernel function  $k(x, y)$ ,  $x \in X_{\text{ref}}$  and  $y \in X_{\text{tr}}$ . The points in feature space are projected onto the eigenvectors  $V^l$  by the formula

$$\chi_{\text{ref}} = K_{\text{ref}} V^l, \quad (25)$$

where  $\chi_{\text{ref}}$  is the low-dimensional projection of the reference points. The test points and reference points in feature space are projected onto the same eigenvectors (see (24)), so the test points and reference points will thus be found in the same  $l$ -dimensional subspace.

The test points and reference points are assumed to be centered in original space, in the sense that

$$\frac{1}{n_{\text{tt}} + n_{\text{ref}}} \left( \sum_{i=1}^{n_{\text{tt}}} x_{\text{tt},i} + \sum_{j=1}^{n_{\text{ref}}} x_{\text{ref},j} \right) = 0. \quad (26)$$

In order to apply the nearest neighbor method, the projected points  $\chi_{\text{tt}}$  and  $\chi_{\text{ref}}$ , and the label set  $y_{\text{ref}}$  are required. The geometric distance between all test points and reference points can be expressed by a distance matrix  $D$ , with its elements given by

$$D_{ij} = d(\chi_i, \chi_j) = \|\chi_i - \chi_j\|_l, \quad (27)$$

where  $\chi_i \in \chi_{\text{tt}}$  and  $\chi_j \in \chi_{\text{ref}}$ .<sup>2</sup> Note that  $\chi_i \in R^l$ , in which  $l$  is the number of eigenvectors that are extracted from the (training) kernel matrix (see (16)). The reference point that is nearest to the  $i$ th test point is then given by the index of the smallest element in the  $i$ th row vector of the matrix  $D$ . In other words, the index  $j_i^*$  of the reference point that is nearest to the  $i$ th test point is given by

$$j_i^* = \underset{j}{\operatorname{argmin}} D_{ij}, \quad j \in \{1, \dots, n_{\text{ref}}\} \quad (28)$$

for  $i = 1, \dots, n_{\text{tt}}$ .<sup>3</sup> In this way, the  $i$ th test point  $x_i$  in  $X_{\text{tt}}$  receives the same label as  $y_{\text{ref}, j_i^*}$ , indicating that it belongs to the same class as the  $j_i^*$ th point in the reference set  $X_{\text{ref}}$ . By performing this procedure on each test point, the whole test set is labeled.

The nearest neighbor method can be extended into taking into account more than one nearest point. The method, referred to as the “ $k$ -nearest neighbor method”, uses a classification function based on the  $k$  nearest points to determine the label of each test point.

#### 4.4 Kernel Regression

As described in [11], kernel regression can be used to form a classification function for the labeling of a partially labeled set. This is done by using a weight-vector

$$c = \{c_1, \dots, c_{n_{\text{tr}}}\} \quad (29)$$

to fit a training kernel matrix  $K_{\text{tr}}$  to the corresponding set of labels  $y = \{y_1, \dots, y_{n_{\text{tr}}}\}$ . Thus, the classification function is trained on data from the training set in order to be able to predict the labels of the data in the test set. In this section, binary classification is considered, meaning that each label can be either  $+1$  or  $-1$ , that is to say,  $y_i \in \{+1, -1\}$ , for  $i = 1, \dots, n_{\text{tr}}$ . The task is to find a vector  $c$  such that,

$$(n_{\text{tr}}\gamma I_{n_{\text{tr}}} + K_{\text{tr}})c = y, \quad (30)$$

where  $I_{n_{\text{tr}}}$  is an  $n_{\text{tr}} \times n_{\text{tr}}$  identity matrix,  $\gamma$  is a parameter. The term  $n_{\text{tr}}\gamma I_{n_{\text{tr}}}$  in (30) is used to make the problem well-posed, meaning that there exists a unique solution

<sup>2</sup>Definition:  $\|x\|_p \triangleq (\sum_{i=1}^d |x_i|^p)^{1/p}$ , where  $p \geq 0$ .

<sup>3</sup>Definition:  $\operatorname{argmin}_x f(x) \triangleq \{x | \forall y : f(x) \leq f(y)\}$ .

to the problem and that there is a continuous relation between the input data and the solution. The parameter  $\gamma$  controls the generalization of the predictions. A “large” value of  $m\gamma$  is said to yield a good condition number, so that minor variations in the test set do not affect the outcome of the labeling. The method is derived from Tikhonov regularization and is further described in [11].

The weight vector  $c$  can then be used to form a classification function  $f : R^{n_{tr}} \rightarrow R$  that is defined as

$$f(x) \triangleq \sum_{i=1}^{n_{tr}} c_i K_{tt, x_i}(x), \quad (31)$$

in which  $K_{tt, x_i}(x)$  is the row vector in the test kernel matrix that corresponds to the test point  $x_i$ . For each test point  $x_i$ , the classification function gives either a positive or negative value. If the value is positive, then  $x_i$  belongs to the class  $+1$ , and if the value is negative, then  $x_i$  belongs to the class  $-1$ . In other words, the classification function labels the test point  $x_i$  as

$$\text{sgn}(f(x_i)) = \begin{cases} +1 & \text{if } f(x_i) \geq 0 \\ -1 & \text{if } f(x_i) < 0. \end{cases} \quad (32)$$

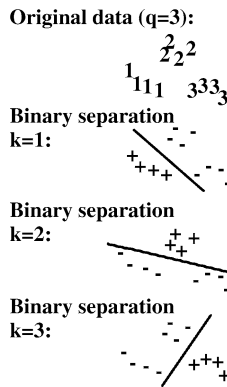
As has been seen, kernel regression introduces the additional parameter  $\gamma$ . There are several techniques for finding a suitable value of  $\gamma$ . The most commonly used is through  $k$ -fold cross-validation. Cross-validation has been the approach of this paper as well.

## 4.5 Multiclass Classification

All the data in the experiments have been multiclass data, in the sense that data points of more than two classes have been involved. Previous experiments have suggested that kernel PCA is more efficient at separating binary class problems than multiclass-problems. Because of this, a variant of the skewed binary classification tree (BCT) approach has been taken for the kernel PCA method [7]. The BCT can be used to classify a multi-class problem, using repeated binary classifications.

Consider the previously described transductive problem in which we have a training set  $X_{tr}$  consisting of  $n_{tr}$  points and a test set  $X_{tt}$  consisting of  $n_{tt}$  points. The labels  $Y^{tr} = \{y_1^{tr}, \dots, y_{n_{tr}}^{tr}\}$  of the points in the training set are known. The task is to determine the labels  $Y^{tt} = \{y_1^{tt}, \dots, y_{n_{tt}}^{tt}\}$  of the points in the test set. Each point in the two sets belong to one of  $q$  classes ( $q > 2$ ). In the method described below, two matrices  $Z^{tr} \in R^{n_{tr} \times q}$  and  $Z^{tt} \in R^{n_{tt} \times q}$  are introduced. Each element in  $Z^{tr}$  and  $Z^{tt}$  can take the value  $+1$  or  $-1$ . The  $i$ th column vector of  $Z^{tr}$  and  $Z^{tt}$  represent the binary labeling of the training set and test set in the case when the  $i$ th class is separated from the other  $q - 1$  classes.





**Fig. 4** Multiclass classification using repeated binary classification. The *numbers* 1, 2 and 3 represent test points that belong to the respective class. The *symbol* “+” represents a point belonging to the class +1 and the symbol “-” represents a point that belongs to the class -1. The *lines* represent the separation of test points (according to class) that results, for example, from kernel regression

In order to perform binary classification on the multi-class classification problem, the following approach was taken:

1. Let  $k = 1$ .
2. For each data point  $x_i \in X_{tr}$ : If  $y_i^{tr} = k$ , then let  $Z_{ik}^{tr} = +1$ . Otherwise, let  $Z_{ik}^{tr} = -1$ .
3. Use kernel regression or the nearest method to assign binary labels ( $Z_{jk}^{tt}$ ,  $j = 1, \dots, n_{tt}$ ) to the testing set, using the new labels  $Z_{ik}^{tr}$ ,  $i = 1, \dots, n_{tr}$  as input for the labeling method.
4. If  $k < q$ , then increase  $k$  by 1, and return to Step 2. Otherwise, continue to the next step.
5. For each data point  $x_j \in X_{tr}$ : Iterate through the classes  $k = 1, \dots, q$ . For the first  $k$  for which  $Z_{jk}^{tr} = 1$ , let  $y_j^{tt} = k$  and go to the next data point. If no such  $k$  exists, then let  $y_j^{tt} = q$ .

In Step 5, the binary label information in  $Z^{tt}$  is transformed into the original multi-class labeling. This is done by the majority voting procedure (see, e.g., [9]). The approach taken in the experiments of this article differs from the standard skewed BCT in that the points that have been separated in one iteration  $k$  are not removed before the following iteration. This is due to evaluation purposes of the classification methods that were used; a way of seeing if any point is assigned more than one label. The advantage of using the original BCT is that the computation time is shorter. See Fig. 4 for an example of multiclass classification using repeated binary classification.

**Table 1** Prediction accuracy of kernel regression (KR) and the nearest neighbor method (NN) applied to Iris data

$\frac{n_{tr}}{n}$ (%)	$\frac{n_{tt}}{n}$ (%)	#Eigenvecs.	$\sigma$	$\gamma$	KR (%)	NN (%)
10	90	5	0.4	1.0000	82.7	92.7
20	80	5	0.4	0.0005	96.8	94.5
20	80	10	0.4	0.0002	94.5	93.1
40	60	5	0.3	0.0001	95.1	94.6
40	60	10	0.3	0.0004	97.6	95.4
40	60	25	0.4	0.0001	95.2	95.3
60	40	10	0.3	0.0005	97.8	96.2
60	40	5	0.3	0.0001	95.3	95.5
90	10	10	0.3	0.0001	96.7	96.7

5 Numerical Experiments

5.1 Kernel PCA Applied to Iris Data

Table 1 presents result of tests in which the nearest neighbor method and kernel regression are tested on an Iris database.<sup>4</sup> The Iris data consist of  $n = 150$  four-dimensional data points, divided into three equally-sized classes. The  $n_{tr}$  data points in the training set were randomly selected from the Iris data. The remaining points were then used as test data. In Table 1, each prediction accuracy-value is an average of 10 sequential tests, in which the training set and test set were randomly selected anew each time.

As can be seen in Table 1, the nearest neighbor method and kernel regression perform about as well when a good value for the parameter  $\gamma$  is found (in this test  $\gamma \ll 1$  gives a good classification performance). The performance of the nearest neighbor method is only dependent on  $\sigma$ . Note that  $\frac{n_{tr}}{n}$  and  $\frac{n_{tt}}{n}$  indicate the percentage of training points and test points, respectively.

5.2 Kernel PCA Applied to Sugar Data

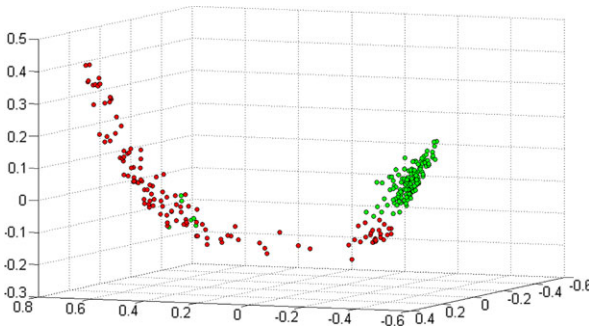
The sugar database consists of 681 eleven-dimensional data points. The sugar data is consists of four different classes: Allose, Galactose, Glucose and Mannose. In the tests on which the results of Table 2 is based,  $\sigma$  had a constant value ( $\sigma = 0.001$ ). The value of  $\sigma$  was found by testing a range of different values. Ten eigenvectors were extracted from the kernel matrix in each test. The percentage of training points in the experiment was 10 %, 20 % and 40 %, respectively. In the table, the prediction accuracies presented on each row is the computed average of 10 tests.

<sup>4</sup> Available online: <http://archive.ics.uci.edu/ml/>.

**Table 2** Prediction accuracy of kernel PCA applied to sugar data

$\frac{n_{tr}}{n}$ (%)	$\frac{n_{tt}}{n}$ (%)	$\gamma$	KR (%)	NN (%)
10	90	0.00027	73.2	64.2
20	80	0.00006	78.6	71.4
40	60	0.00002	79.8	75.5
Average:			77.2	70.3

**Fig. 5** Separation of Allose and Mannose



In this test, the kernel regression method yields higher classification accuracies than the nearest neighbor method. However, as the percentage of training points increased, the performance of the nearest neighbor method improved to levels close to that of kernel regression.

Most of the predictions error in the numerical experiments are due to misclassifications of points belonging to the classes Allose and Mannose. Experiments were performed in which these two classes were treated separately—see Fig. 5. Using KPCA and kernel regression and 20 % of the data as training points, an average prediction accuracy of 94.0 % was achieved for the two classes. The value of the parameters were  $\sigma = 0.15$  and  $\gamma = 0.000006$  and each result was the average prediction accuracy of 20 tests. Using 80 % and 99 % training points, the prediction accuracy increased to 97.5 % and 98.3 %, respectively.

5.3 Using a Modified Labeling Function

As described in Sect. 4.4, the binary labels can be determined in kernel regression by the sign of the elements of the classification function  $f(x)$ , so that the  $i$ th point  $x_i$  is labeled as  $\text{sgn}(f(x_i))$ . A possible way of improving the classification performance of certain data is to use a modified version of (32), namely the function

$$\text{sgn}(f(x) + \delta), \quad \delta \in R. \tag{33}$$

The parameter  $\delta$  can be used to shift the vector of label values  $f(x)$  in one direction, increasing the likelihood of a +1 labeling ( $\delta > 0$ ) or −1 labeling ( $\delta < 0$ ).

**Table 3** Prediction accuracy of kernel regression, using the modified labeling function

$\frac{n_{tr}}{n} (%)$	KR ( $\delta = 0$ )	KR ( $\delta = -1/3$ )
20	77.0	83.1
40	79.5	85.6
60	81.6	86.9

**Table 4** A comparison of PCA and KPCA, applied to sugar data

$\frac{n_{tr}}{n} (%)$	$\delta$	PCA (%)	KPCA (%)
20	0	67.4	76.7
20	$-1/3$	77.3	83.3

It was found that  $\delta = -1/3$  gave a noticeable increase of prediction accuracy for the sugar data (see Table 3). In these tests, the parameters were set to  $\sigma = 0.0006$  and  $\gamma = 0.000006$ .

It is interesting to see how kernel PCA performs compared to the linear PCA method. In order to perform standard PCA the kernel function

$$k(x, y) = (x^T \cdot y) \tag{34}$$

was used, where the points  $x$  and  $y$  are represented as row vectors. The results in Table 4 suggest that for the sugar data, kernel PCA has a somewhat higher prediction accuracy than standard PCA. The function in (33), with  $\delta = -1/3$ , notably increases the performance of both methods.

Results presented in this section are generally the computed average of 20 or 25 tests (see the [Appendix](#) for detailed numerical results).

**5.4 Using Label-Based Weights**

One way of improving the separation of data points belonging to different classes is to modify the kernel matrix by using the label information that is available. There are different approaches for doing this, see, for example, [8]. In the tests that are described in this section, a factor, proportional to the difference in label values, was inserted into the kernel function. Two different cases are considered. In the first, the labels of the whole data set are known and used for modifying the kernel matrix. In the second case, only the training set is assumed to be known and used to modify the kernel matrix.

The RBF kernel function, that was given in (17), is modified by inserting a weight  $\mu_{ij}$ , so that the elements of the kernel matrix are given by

$$K_{ij} = \exp(-\mu_{ij}\sigma \|x_i - x_j\|^2), \quad (35)$$

where the weight  $\mu_{ij}$  is determined by

$$\mu_{ij} = \mu(y_i, y_j) = (y_i - y_j)^2. \quad (36)$$

Note that the function will then take the values

$$\mu_{ij} = \begin{cases} 0 & \text{if } y_i = y_j \\ > 0 & \text{if } y_i \neq y_j. \end{cases} \quad (37)$$

This way, if two points have the same label, then their inner-product in  $F$  will be one ( $\mu_{ij} = 0$ ), meaning that the vectors representing the two points are parallel. On the other hand, if the labels are different, then the inner product between the two points in feature space will decrease or remain the same (depending on the labels), which is equivalent of the angle between the vectors of the two points becoming more right-angled.

The figures presented in this section show the projection of the test data points onto the three Eigenvectors that captures the largest amount of the variance, that is, with the largest corresponding Eigenvalues.

Figure 6 shows the separation of sugar data using the weights based on label-information that was described above for both the training kernel matrix and test kernel matrix. Likewise, Fig. 7 shows the separation of the Iris data. This technique separates the data points very well, in the sense that the data points of each class are clustered together and the points of each class is distinctively separated from other points.

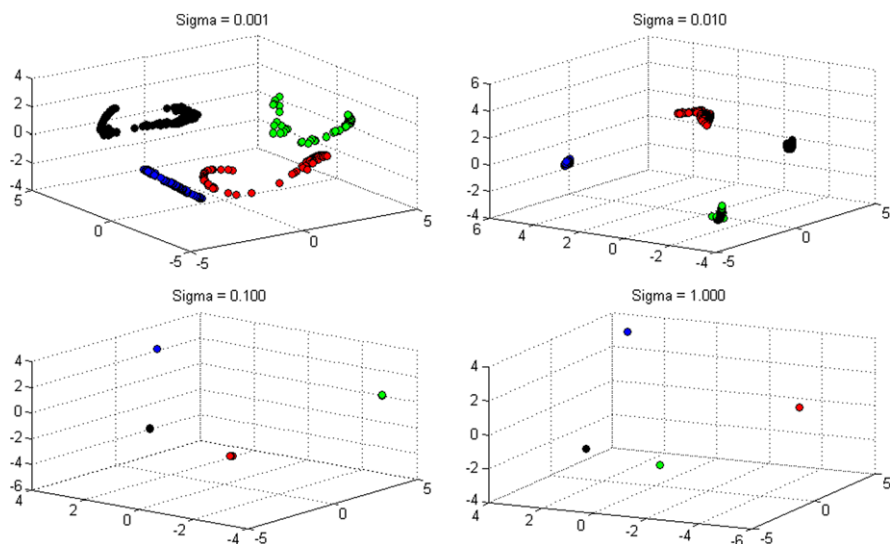
Note that the kernel becomes more discriminatory for larger values of  $\sigma$ , resulting in an increased distance between the clusters of points.

Previously the transduction problem has been considered, in which the labels of the training set are known and those of the test set are unknown. However, in the results of Figs. 6 and 7, label information for both the training and test set has been used to compute the training kernel matrix and test kernel matrix. Thus, these are not to be considered transduction problems.

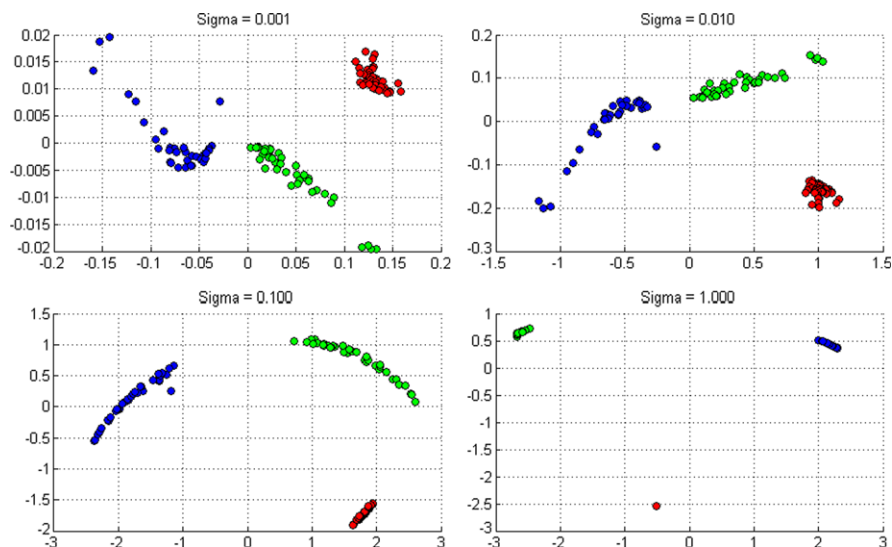
Yet, the method of using weights based on label-information can be used in a transductive setting as well. This can be done by using the labels of the training set to compute the training kernel matrix. Figure 8 shows the Iris data when the weights are only applied to the training kernel matrix. Figure 9 shows the impact of weights on the Sugar data. On the left side of the figure, the weights are used.

## 6 Discussion

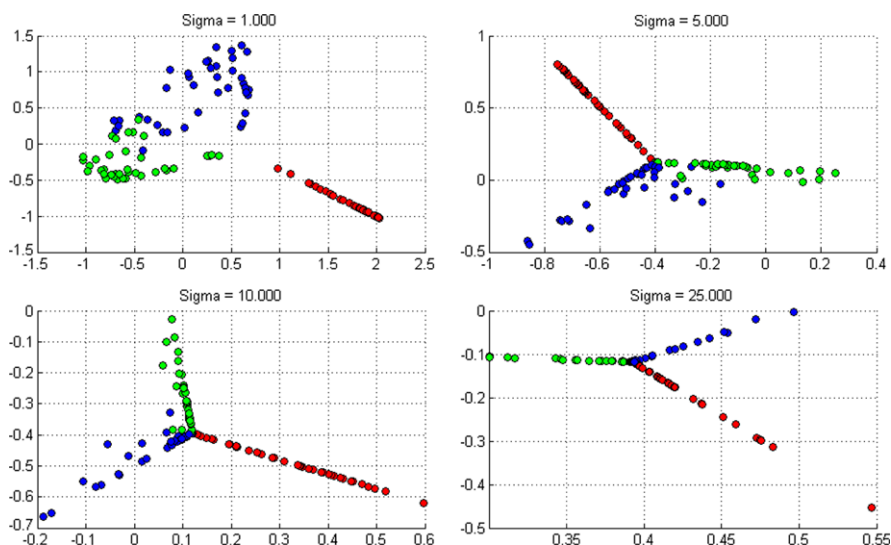
From the experiments it appears that for these databases, kernel regression performs better in terms of prediction accuracy than the nearest neighbor method. However,



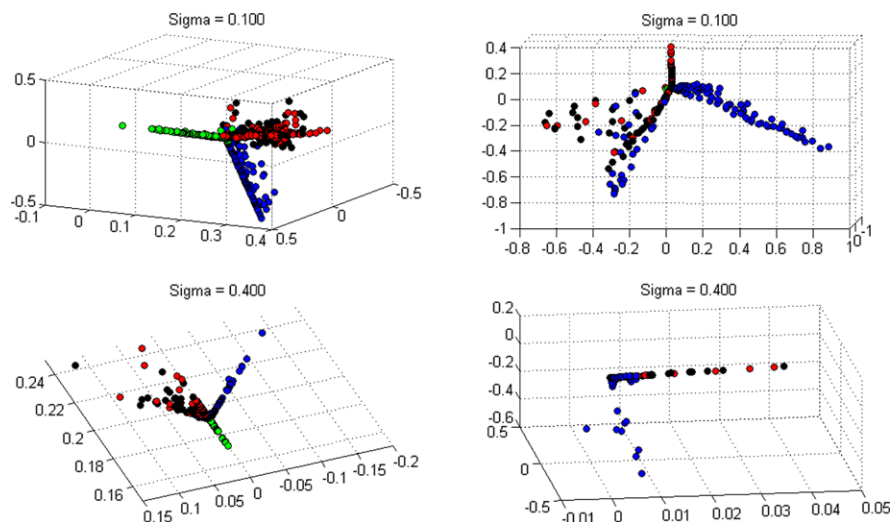
**Fig. 6** Test points of label-based weights in which labels of all data points are used, applied to Sugar data. The sugar classes in the plot are Allose (*red*), Galactose (*green*), Glucose (*blue*) and Mannose (*black*). The four plots show the clustering of points for different  $\sigma$ . In all four classes, the separations are good enough to linearly separate the points according to class, and for  $\sigma = 0.1$  and  $\sigma = 1$  the separation is nearly perfect (Color figure online)



**Fig. 7** Test points of label-based weights in which labels of all data points are used, applied to Iris data. The plots show the separations of classes for different values of  $\sigma$ . In all cases, it is possible to linearly separate the three classes from each other. For  $\sigma = 1$ , the separation is nearly perfect, in the sense that the points of each class are highly concentrated in clusters



**Fig. 8** Test points of Iris data, when label-based weights are applied to the training kernel matrix (only labels of the training points are used). The four plots present the resulting separation when different values of  $\sigma$  are used. Increased  $\sigma$ -values resulted in better separations



**Fig. 9** Test points of Sugar data, with label-based weights (*left*) and without weights (*right*). Only the labels of the training points are used in the weights. The four plots show the separation of the sugar classes for different values of  $\sigma$ . The sugar classes are Allose (*red*), Galactose (*green*), Glucose (*blue*) and Mannose (*black*). As can be seen from the plots, Allose and Mannose are most difficult to separate (and interestingly, chemically more alike) (Color figure online)

using kernel regression comes with the cost of introducing an additional parameter  $\gamma$ .

The highest average prediction accuracies that were achieved using 20 % of the set as training points were 97.5 % for the Iris data and 83 % for the Sugar data. Using 60 % of the data as training data, the prediction accuracy for the Sugar data increased to around 87 %.

The comparisons between kernel PCA and standard PCA showed that kernel PCA consistently gave higher prediction accuracies for the databases mentioned.

**Acknowledgements** The main part of this work was done during the visit of the first named author at the Center for Applied Optimization, University of Florida, Gainesville, whose hospitality is greatly acknowledged.

## Appendix: Detailed Numerical Results

This section gives the detailed numerical results of the experiments that were presented in Sect. 5. All experiments were performed with randomly selected training sets. The sugar database is used in all the tests of this section.

### A.1 Comparison of PCA and Kernel PCA

In these experiments, kernel PCA was compared to standard PCA. The two methods were applied to sugar data. Table 5 shows the results.

The following parameters were used in the experiments:

- PCA20: 20 % training points,  $\delta = 0$ .
- dPCA20: 20 % training points,  $\delta = -1/3$ .
- KPCA20: 20 % training points,  $\sigma = 0.0006$ ,  $\gamma = 0.000006$ ,  $\delta = 0$ .
- dKPCA20: 20 % training points,  $\sigma = 0.0006$ ,  $\gamma = 0.000006$ ,  $\delta = -1/3$ .

### A.2 Comparison of $\delta = 0$ and $\delta = -1/3$

These tests were performed to measure the impact that the parameter  $\delta$  had on the prediction accuracy in kernel regression. Several other values of  $\delta$  were tried as well (both positive and negative), but are not included in this report.  $\delta = -1/3$  appeared to be optimal for the sugar data. The results are shown in Table 6. Also, some tests were performed in which kernel regression with noncentered kernel matrices were used. The following parameters were used in the experiments:



**Table 5** Prediction accuracies of PCA and KPCA applied to sugar data

Nr.	PCA20	dPCA20	KPCA20	dKPCA20
1	66.20	80.00	76.00	86.80
2	68.40	80.90	76.00	82.80
3	63.70	78.40	77.40	84.20
4	67.30	79.50	77.40	85.90
5	67.50	74.50	76.50	83.10
6	68.80	73.00	78.50	83.90
7	65.30	78.00	82.40	83.30
8	69.40	80.90	76.20	82.80
9	68.10	74.10	79.60	77.10
10	67.90	78.50	78.20	79.10
11	70.60	75.40	76.50	84.40
12	66.40	79.30	71.70	86.20
13	68.80	77.60	75.80	84.20
14	64.20	78.70	75.10	84.20
15	68.60	76.50	76.20	83.70
16	67.70	76.30	73.00	80.40
17	64.40	77.10	78.50	84.80
18	65.90	74.10	72.30	84.00
19	71.40	77.30	77.80	84.40
20	68.80	79.30	73.40	81.30
21	67.70	80.40	79.10	80.90
22	66.40	71.70	75.20	83.30
23	65.70	76.50	78.40	84.00
24	65.90	74.50	77.30	82.90
25	68.60	80.70	78.20	85.10
Average:	67.40	77.30	76.70	83.30

- KPCA20c: 20 % training points.  $\sigma = 0.25$ ,  $\gamma = 0.7$ ,  $\delta = 0$ . In these tests a non-centered kernel matrix is used.
- KPCA40: 40 % training points.  $\sigma = 0.0006$ ,  $\gamma = 0.000006$ ,  $\delta = 0$ .
- dKPCA40: 40 % training points.  $\sigma = 0.0006$ ,  $\gamma = 0.000006$ ,  $\delta = -1/3$ .
- KPCA60: 60 % training points.  $\sigma = 0.0006$ ,  $\gamma = 0.000006$ ,  $\delta = 0$ .
- dKPCA60: 60 % training points.  $\sigma = 0.0006$ ,  $\gamma = 0.000006$ ,  $\delta = -1/3$ .

**Table 6** Comparison of prediction accuracies using  $\delta = 0$  and  $\delta = -1/3$ 

Nr.	KPCA20c	KPCA40	dKPCA40	KPCA60	dKPCA60
1	79.80	79.70	87.50	82.00	86.80
2	79.30	79.70	86.30	81.30	87.90
3	80.70	78.70	85.10	82.70	88.20
4	80.90	82.20	87.30	80.20	83.80
5	78.90	80.90	85.10	84.60	85.30
6	80.70	80.90	86.60	80.50	90.40
7	83.50	80.00	85.30	84.60	90.80
8	79.80	81.70	87.00	79.40	85.30
9	80.90	80.00	83.90	79.80	87.90
10	83.50	80.00	85.80	80.90	88.60
11	79.80	79.70	86.60	82.40	85.30
12	81.30	79.70	84.10	80.20	86.00
13	83.30	75.80	85.80	81.60	86.40
14	81.80	78.70	84.10	79.40	83.50
15	79.50	79.70	88.00	76.10	86.40
16	83.30	79.20	84.60	84.60	84.20
17	82.80	78.50	81.40	84.60	83.80
18	83.70	79.20	85.10	82.40	84.20
19	80.60	82.60	86.60	81.30	88.60
20	80.00	76.50	84.60	82.40	86.80
21	79.60	77.00	85.10	82.70	88.20
22	81.80	80.70	86.30	80.90	88.60
23	79.60	76.80	86.80	80.90	90.40
24	82.60	79.20	87.00	83.80	87.10
25	83.10	80.00	84.10	80.20	87.10
Average:	81.20	79.50	85.60	81.60	86.90

## References

1. Achlioptas, D.: Database-friendly random projections. In: Proc. ACM Symp. on the Principles of Database Systems, pp. 274–281 (2001)
2. Aronszajn, N.: Theory of Reproducing Kernels. Defense Technical Information Center, Harvard University, Cambridge (1950)
3. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001), pp. 245–250 (2001)
4. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**, 21–27 (1967)
5. Cucker, F., Smale, S.: On the mathematical foundations of learning. Bull. Am. Math. Soc. **39**(1), 1–50 (2002)
6. Jolliffe, I.T.: Principal Component Analysis. Springer, New York (2002)

7. Lee, J.-S., Oh, I.-S.: Binary classification trees for multi-class classification problems. In: ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition, Washington, DC, USA, p. 770. IEEE Comput. Soc., Los Alamitos (2003)
8. Min, R., Bonner, A., Zhang, Z.: Modifying kernels using label information improves SVM classification performance (2007)
9. Narasimhamurthy, A.: Theoretical bounds of majority voting performance for a binary classification problem. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(12), 1988–1995 (2005)
10. Pardalos, P.M., Hansen, P. (eds.): *Data Mining and Mathematical Programming*. Am. Math. Soc., Providence (2008)
11. Poggio, T., Smale, S.: The mathematics of learning: dealing with data. *Not. Am. Math. Soc.* **50**, 537–544 (2003)
12. Schölkopf, B., Smola, A., Müller, K.-R.: Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.* **10**, 1299–1319 (1998)
13. Shawe-Taylor, J., Cristianini, N.: *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge (2004)

# Distance-Based Clique Relaxations in Networks: $s$ -Clique and $s$ -Club

Shahram Shahinpour and Sergiy Butenko

**Abstract** The concept of the clique, originally introduced as a model of a cohesive subgroup in the context of social network analysis, is a classical model of a cluster in networks. However, the ideal cohesiveness properties guaranteed by the clique definition put limitations on its applicability to situations where enforcing such properties is unnecessary or even prohibitive. Motivated by practical applications of diverse origins, numerous *clique relaxation models*, which are obtained by relaxing certain properties of a clique, have been introduced and studied by researchers representing different fields. *Distance-based* clique relaxations, which replace the requirement on pairwise distances to be equal to 1 in a clique with less restrictive distance bounds, are among the most important such models. This chapter surveys the up-to-date progress made in studying two common distance-based clique relaxation models called  $s$ -clique and  $s$ -club, as well as the corresponding optimization problems.

**Keywords** Clique relaxations ·  $s$ -Clique ·  $s$ -Club · Maximum clique

## 1 Introduction

In 1949, Luce and Perry [42] introduced the clique concept to model the notion of a *cohesive subgroup* in social network analysis. Since then, cliques and the associated maximum clique problem have become ubiquitous and have been studied extensively in graph theory [13, 14, 28], theoretical computer science [32, 39] and operations research [9, 15, 20] from different perspectives. In graph-theoretic terms, a clique is a subset of vertices that are pairwise adjacent. The clique definition ensures the perfect reachability between the group's entities, as they are directly linked to each other. Moreover, it also ensures that a clique has the highest possible degree

---

S. Shahinpour · S. Butenko (✉)

Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77843-3131, USA

e-mail: [butenko@tamu.edu](mailto:butenko@tamu.edu)

S. Shahinpour

e-mail: [shahinpour@tamu.edu](mailto:shahinpour@tamu.edu)

of each vertex, the highest possible connectivity, and the largest possible number of edges in the induced subgraph among all subsets of vertices of the same cardinality. However, the ideal cohesiveness properties of a clique put limitations on its applicability to situations where enforcing such properties is unnecessary or even prohibitive. For example, in transportation and telecommunication networks easy reachability between the members of a group (or a cluster) is of utmost importance, whereas a large number of edges is either costly to construct and maintain or results in operating inefficiencies, such as excessive interference.

To address particular practical aspects that cannot be suitably modeled by cliques, numerous clique relaxation models have been introduced in the literature that enforce certain elementary properties of cliques to be present, in a relaxed form, in the model of a cluster. The long list of the proposed models includes the distance-based clique relaxations called  $s$ -clique [41] and  $s$ -club [49], degree-based relaxations called  $s$ -plex [62] and  $k$ -core [61], and an edge density-based model known as quasi-clique [1] among many others. The focus of this chapter is on distance-based clique relaxations,  $s$ -clique and  $s$ -club.

Originally proposed by Luce [41] in 1950,  $s$ -clique was the historically first clique relaxation concept. This structure relaxes the requirement of having an edge (distance 1) between any pair of vertices from the group by allowing them to be at most distance  $s$  apart, thus ensuring that they can communicate via a path of at most  $s - 1$  intermediate vertices. Note that these intermediate vertices, while guaranteeing the reachability in at most  $s$  hops between vertices from an  $s$ -clique, do not have to be a part of the  $s$ -clique themselves, which may be considered a drawback from the cohesiveness standpoint. This was first pointed out by Alba [3], who proposed a definition of the so-called *sociometric clique of diameter  $s$* , which was later refined by Mokken [49] under the name of  $s$ -club. Any two members of an  $s$ -club are required to be connected by a path of length at most  $s$ , where all intermediate vertices belong to the  $s$ -club.

The objective of this chapter is to provide an up-to-date survey of known results concerning  $s$ -clique,  $s$ -club, and the corresponding optimization problems, as well as to identify related open questions for future work. The remainder of the chapter is organized as follows. We start by introducing the formal definitions and notations used throughout the chapter in Sect. 2. In Sect. 3, we discuss some basic structural properties of  $s$ -cliques and  $s$ -clubs and the complexity results for the optimization problems associated with these models in order to have a better understanding of the computational challenges one has to overcome in order to solve the problems of interest. Section 4 provides integer programming formulations proposed for the optimization problems. Known polyhedral combinatorics results associated with these formulations are reviewed in Sect. 5. An overview on the solution methods that have been proposed for these problems, along with a brief review of the computational results, is presented in Sect. 6. Selected applications of the problems of interest are discussed in Sect. 7. Finally, the article concludes by discussing some possible research directions for future work in Sect. 8.

## 2 Definitions and Notations

We consider a simple undirected graph  $G = (V, E)$  with the set of vertices  $V$  and the set of edges  $E$  corresponding to pairs of vertices. Two vertices  $v$  and  $v'$  in  $G$  are said to be *adjacent* or *neighbors* if  $(v, v') \in E$ , in which case the edge  $(v, v')$  is said to be *incident* to  $v$  and  $v'$ . Let  $N_G(v) = \{v' \in V : (v, v') \in E\}$  denote the *neighborhood* of a vertex  $v$  in  $G$ , and let  $N_G[v] = \{v\} \cup N_G(v)$  be the *closed neighborhood* of  $v$ . The cardinality of the neighborhood,  $|N_G(v)|$ , is called the degree of  $v$  in  $G$  and is denoted by  $\deg_G(v)$ . Let  $\delta(G)$  and  $\Delta(G)$  denote the minimum and the maximum degree of a vertex in  $G$ , respectively. We call a graph  $G' = (V', E')$  a *subgraph* of  $G = (V, E)$  if  $V' \subseteq V$  and  $E' \subseteq E$ . For a subset of vertices  $S \subseteq V$ , the *subgraph induced by  $S$* ,  $G[S]$ , is given by  $G[S] = (S, E \cap (S \times S))$ , where “ $\times$ ” denotes the Cartesian product.

A *path* of length  $r$  between vertices  $v$  and  $v'$  in  $G$  is a subgraph of  $G$  given by an alternating sequence of distinct vertices and edges  $v \equiv v_0, e_0, v_1, e_1, \dots, v_{r-1}, e_{r-1}, v_r \equiv v'$  such that  $e_i = (v_i, v_{i+1}) \in E$  for all  $1 \leq i \leq r-1$ . A cycle of length  $r$  is defined similarly, by assuming that  $v \equiv v'$  in the definition of a path. If there is at least one path between two vertices  $v$  and  $v'$  in  $G$ , then we say that  $v$  and  $v'$  are *connected* in  $G$ . A graph is called *connected* if any pair of its vertices is connected. Otherwise, a graph is called *disconnected*. The length of a shortest path between two connected vertices  $v$  and  $v'$  in  $G$  is called the *distance* between  $v$  and  $v'$  in  $G$  and is denoted by  $d_G(v, v')$ . If  $v$  and  $v'$  are not connected in  $G$ , then  $d_G(v, v') = \infty$ . The diameter  $\text{diam}(G)$  of a graph  $G$  is given by the maximum distance between any pair of vertices in  $G$ , that is,  $\text{diam}(G) = \max_{v, v' \in V} d_G(v, v')$ .

The *vertex connectivity*  $\kappa(G)$  of  $G$  is the minimum number of vertices that need to be deleted from  $G$  in order to obtain a disconnected or a trivial graph. The *density*  $\rho(G)$  of  $G$  is given by  $\rho(G) = |E|/\binom{|V|}{2}$ . A *complete graph*  $K_n$  on  $n$  vertices is a graph that contains all possible edges, that is,  $\rho(K_n) = 1$ . The *complement*  $\bar{G}$  of  $G$  is  $\bar{G} = (V, \bar{E})$ , where  $\bar{E}$  is the complement of  $E$ , that is,  $E \cap \bar{E} = \emptyset$  and  $K_{|V|} = (V, E \cup \bar{E})$ . A *clique*  $C$  is a subset of vertices such that  $G[C]$  is a complete graph. An independent set  $I$  is a subset of vertices such that  $G[I]$  has no edges. Clearly,  $S \subseteq V$  is a clique in  $G$  if and only if  $S$  is an independent set in  $\bar{G}$ . A clique (independent set) is called *maximal* if it is not a subset of a larger clique (independent set). A *maximum clique* (*independent set*) of  $G$  is a clique (independent set) of the largest size in  $G$ , and the problem of finding a maximum clique (independent set) in a graph is called the *maximum clique* (*independent set*) *problem*. The size of a maximum clique in  $G$  is called the *clique number* of  $G$  and is denoted by  $\omega(G)$ . The size of a maximum independent set in  $G$  is called the *independence number* of  $G$  and is denoted by  $\alpha(G)$ . We have  $\omega(G) = \alpha(\bar{G})$ .

Some of the well known clique relaxation models are defined next. We assume that  $s$  and  $k$  are positive integer constants and  $\lambda, \gamma \in (0, 1]$  are real constants. Let  $S \subseteq V$ .  $S$  is an  *$s$ -plex* if  $\delta(G[S]) \geq |S| - s$ .  $S$  is an  *$s$ -defective clique* if  $G[S]$  contains at least  $\binom{|S|}{2} - s$  edges.  $S$  is a  *$k$ -core* if  $\delta(G[S]) \geq k$ .  $S$  is a  *$k$ -block* if  $\kappa(G[S]) \geq k$ .  $S$  is a  *$\gamma$ -quasi-clique* if  $\rho(G[S]) \geq \gamma$ .  $S$  is a  *$(\lambda, \gamma)$ -quasi-clique* if  $\delta(G[S]) \geq \lambda(|S| - 1)$  and  $\rho(G[S]) \geq \gamma$ .  $D \subseteq V$  is called a *distance  $s$ -dominating*

set if for any  $v \in V \setminus D$  there exists  $v' \in D$  such that  $d_G(v, v') \leq s$ . Distance 1-dominating set is called simply a *dominating set*.

In [54], the motivation behind some of the most popular clique relaxation models was analyzed in a systematic fashion, and a set of simple rules for defining meaningful clique relaxation structures was identified, yielding a methodical taxonomic framework for clique relaxations. The framework is based on the observation that the clique can be defined using alternative equivalent descriptions via other basic graph concepts, such as distance, diameter, domination, degree, density, and connectivity. The corresponding equivalent definitions are referred to as *elementary clique defining properties*. Then, by applying some simple modifications to the elementary clique defining properties, one can reproduce the known clique relaxation models, as well as define new structures of potential practical interest. We will adhere to this framework in defining the distance-based clique relaxations formally as follows.

First, note that a subset of vertices  $C$  is a clique in  $G$  if and only if  $d_G(v, v') = 1$ , for any  $v, v' \in C$  or, equivalently,  $\text{diam}(G[C]) = 1$ . These equivalent clique definitions constitute the elementary clique defining properties based on distance and diameter, respectively. In both cases, we have an equivalent characterization of a clique by setting a certain parameter (pairwise distance or diameter) to its minimum possible value. We can define the corresponding clique relaxations by restricting the violation of the respective elementary clique defining property, that is, by allowing the pairwise distance or diameter to be greater than 1, but no greater than a constant positive integer  $s > 1$ . As a result, we obtain the following definitions.

**Definition 1** ( $s$ -clique) Given a simple undirected graph  $G = (V, E)$  and a positive integer constant  $s$ , a subset of vertices  $S \subseteq V$  is called an  $s$ -clique if  $d_G(v, v') \leq s$ , for any  $v, v' \in S$ .

**Definition 2** ( $s$ -club) Given a simple undirected graph  $G = (V, E)$  and a positive integer constant  $s$ , a subset of vertices  $S \subseteq V$  is called an  $s$ -club if  $\text{diam}(G[S]) \leq s$ .

An  $s$ -clique ( $s$ -club) is called maximal in  $G$  if it is not a subset of a larger  $s$ -clique ( $s$ -club) in  $G$ , and maximum in  $G$  if there is no larger  $s$ -clique ( $s$ -club) in  $G$ . The maximum  $s$ -clique ( $s$ -club) problem asks to find a maximum  $s$ -clique ( $s$ -club) in  $G$ . The size of the largest  $s$ -clique in  $G$  is called the  $s$ -clique number and is denoted by  $\tilde{\omega}_s(G)$ . The size of the largest  $s$ -club in  $G$  is called the  $s$ -club number and is denoted by  $\overline{\omega}_s(G)$ .

According to the taxonomy in [54], clique relaxations based on restricting the violation of an elementary clique defining property can be standard or weak; absolute or relative; and structural or statistical. For a standard relaxation, we require the relaxed clique-defining property to hold in the *induced subgraph*, whereas the corresponding weak relaxation requires the same property to be satisfied within the *original graph* instead of the induced subgraph. Since  $s$ -clique is defined by restricting pairwise distances for its members in  $G$ , it is a weak relaxation, whereas  $s$ -club, which restricts distances in the induced subgraph, is a standard relaxation. Both  $s$ -clique and  $s$ -club are absolute relaxations, since the value of the constant  $s$

refers to the absolute bound on the distance in  $G$  or  $G[S]$  and does not depend on the size of  $S$ . However, their relative version could easily be introduced by replacing the constant  $s$  in the definitions of  $s$ -clique and  $s$ -club with  $\gamma|S|$ , where  $\gamma \in (0, 1)$  is a constant. Finally, both  $s$ -clique and  $s$ -club are structural clique relaxations and their statistical counterparts could be defined by requiring that the *average* pairwise distance between vertices for  $S$  in  $G$  or  $G[S]$  is at most  $s$ . It should be noted that, in contrast to the structural relaxations, statistical relaxations generally impose little in terms of the group structure.

Higher-order clique relaxation models, which relax more than one elementary clique defining properties simultaneously, could also be defined using distance or diameter restrictions in addition to other requirements. Since the graph-theoretic notion of distance relies on paths, in addition to the simple higher order relaxations that combine multiple properties in a straightforward fashion,  $s$ -clique and  $s$ -club could also be involved in the so-called  $k$ -hereditary higher-order relaxations, with  $k$ -connectivity embedded within their structure. Namely,  $k$ -hereditary  $s$ -club and  $s$ -clique can be defined as follows. Given  $G = (V, E)$  and positive integers  $s$  and  $k$ ,  $S \subseteq V$  is called a  $k$ -hereditary  $s$ -club if  $\text{diam}(G[S \setminus S']) \leq s$  for any  $S' \subset S$  such that  $|S'| \leq k$ . Similarly,  $S$  is a  $k$ -hereditary  $s$ -clique if  $d_G(v, v') \leq s$  for all  $v, v' \in S \setminus S'$  for any  $S' \subset S$  such that  $|S'| \leq k$ .

While the variations of distance-based relaxations just defined may potentially find interesting applications, in the remainder of this chapter, we focus on the  $s$ -clique and  $s$ -club models, which were referred to as *canonical* clique relaxation models for distance and diameter, respectively, in [54].

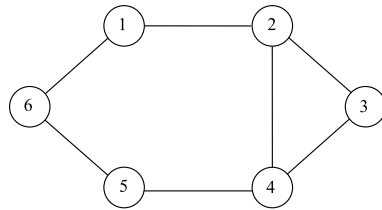
Next, we introduce some graph classes for which the problems of interest have been explored in the literature. Consider a graph  $G = (V, E)$ . Given a cycle in  $G$ , its *chord* is an edge between two vertices of the cycle that is not a part of the cycle. A graph is called *chordal* if any cycle on at least 4 vertices has a chord.  $G$  is called a  $k$ -partite graph if  $V$  can be partitioned into  $k$  non-overlapping independent sets. If  $k = 2$ , a  $k$ -partite graph is *bipartite*.  $G$  is a *split graph*, if  $V = V_1 \cup V_2$ , where  $V_1$  is a clique and  $V_2$  is an independent set such that  $V_1 \cap V_2 = \emptyset$ .  $G$  is an *interval graph* if there exists a set of intervals  $\mathcal{I} = \{I_v : v \in V\}$  on the real line such that  $I_v \cap I_{v'} \neq \emptyset$  iff  $(v, v') \in E$ .

### 3 Structural Properties and Computational Complexity

From the definitions, it is clear that an  $s$ -club is also an  $s$ -clique, however, the converse is not true in general. Even though the  $s$ -clique and  $s$ -club models appear to be very similar, there are some fundamental differences in their structural properties that have important implications for the associated optimization problems. To highlight these differences, consider a simple example in Fig. 1 that originally appeared in [3]. In the graph in this figure, a subset of vertices  $\{1, 2, 3, 4, 5\}$  is a 2-clique, but not a 2-club, since the distance between vertices 1 and 5 is 3 in the induced subgraph. Moreover,  $\{1, 2, 4\}$  is a 2-clique and a 2-club,  $\{1, 2, 4\} \cup \{5\}$  and  $\{1, 2, 4\} \cup \{6\}$



**Fig. 1** A graph illustrating structural differences of 2-cliques and 2-clubs



are both 2-cliques but not 2-clubs, whereas  $\{1, 2, 4\} \cup \{5, 6\}$  is again a 2-clique and a 2-club. This shows the lack of any type of *heredity* for  $s$ -clubs, which is formally defined as follows [54]. A graph property  $\Pi$  is called *hereditary on induced subgraphs*, if for any graph  $G$  with property  $\Pi$  deleting any subset of vertices does not produce a graph violating  $\Pi$ . A graph property  $\Pi$  is called *weakly hereditary*, if for any graph  $G = (V, E)$  with property  $\Pi$  all subsets of  $V$  posses the property  $\Pi$  in  $G$ .

Unlike  $s$ -clubs,  $s$ -cliques posses weak heredity, which allows to reduce the problem of finding an  $s$ -clique to the problem of finding a clique in an auxiliary power graph defined as follows. Given a graph  $G = (V, E)$ , the  $s$ th power of  $G$ , denoted by  $G^s$ , is given by  $G^s = (V, E^s)$ , where  $E^s = \{(v, v') : 0 < d_G(v, v') \leq s\}$ . Then  $S \subseteq V$  is an  $s$ -clique in  $G$  if and only if  $S$  is a clique in  $G^s$ . Heredity on induced subgraphs is the core property implicitly exploited by some of the most successful combinatorial algorithms for the maximum clique problem [21, 51], which can also be applied to  $G^s$  in order to solve the maximum  $s$ -clique problem in  $G$ . Because of the presence of weak heredity,  $s$ -clique has advantage over  $s$ -club in terms of applicability of the variety of existing techniques available for the maximum clique problem to solving the maximum  $s$ -clique problem. However, this comes at a price. The fact that the  $s$ -clique is defined by restricting the distances in the original graph rather than the induced subgraph leads to the possibility of absence of any cohesiveness in the subgraph induced by an  $s$ -clique. For example, the subset of vertices  $\{1, 3, 5\}$  in the graph on Fig. 1 is a 2-clique that induces an independent set, a structure that can hardly be considered cohesive by any standards. In terms of cohesiveness, the worst-case example of an  $s$ -club is a star graph, where one “central” vertex is adjacent to all other vertices, which have no neighbors other than the central vertex. While this structure appears to be quite fragile, as removing the central vertex makes it an independent set, it is still more cohesive than the worst-case example of an  $s$ -clique, which is an independent set to begin with. Since  $s$ -clique does not have to be connected in general, it makes sense to consider a *connected  $s$ -clique*, which is an  $s$ -clique that induces a connected subgraph.

Since  $s$ -clubs do not have any form of heredity defined above, the maximum clique algorithms cannot be easily adapted for the maximum  $s$ -club problem. In fact, the problem of finding a maximal  $s$ -club, which is very easy for clique and  $s$ -clique, becomes challenging. Indeed, the problem of checking whether a given clique ( $s$ -clique) is maximal reduces to checking whether there is a vertex from outside that can be added to the clique ( $s$ -clique). However, the example above clearly shows that this strategy will not work for  $s$ -clubs. In fact, Mahdavi and Balasun-

daram [44] have recently shown that testing whether an  $s$ -club is maximal is NP-hard for any fixed integer  $s \geq 2$ . They have also identified sufficient conditions for every connected 2-clique to be a 2-club based on the concept of a *partitionable cycle*, which can be defined as follows. Consider two nonadjacent vertices  $v$  and  $v'$  in a cycle  $C$  in  $G$ . Removing these two vertices breaks the cycle into two paths,  $P_1(v, v')$  and  $P_2(v, v')$  with the vertex sets  $V_1(v, v')$  and  $V_2(v, v')$ , respectively. If there exist  $v, v'$  such that  $G[V_1(v, v')] = P_1(v, v')$  and  $G[V_2(v, v')] = P_2(v, v')$  then  $C$  is called a partitionable cycle. If, in addition,  $|V_1(v, v')| \neq |V_2(v, v')|$  then the partitionable cycle  $C$  is called asymmetric. Mahdavi and Balasundaram [44] have proved that if no subset of  $5 \leq c \leq 2s + 1$  vertices induces an asymmetric partitionable cycle in  $G$ , where  $s \geq 2$ , then every connected  $s$ -clique is an  $s$ -club. This implies, in particular, that in a bipartite graph every connected 2-clique is a 2-club, which induces a complete bipartite subgraph. In cases where every connected  $s$ -clique is an  $s$ -club, checking maximality of an  $s$ -club reduces to checking maximality of a connected  $s$ -clique and hence is easy. Thus, discovering more of such cases is an interesting future research direction, which will provide further insights towards understanding the complexity of the problem.

The maximum clique problem is a classical NP-hard problem [32, 39], which is also hard to approximate. Recall that for a maximization problem with an optimal objective value given by  $\text{opt}(G)$  on an input graph  $G$ , an algorithm  $\mathcal{A}$  is called  $\sigma$ -approximation algorithm (or algorithm with approximation ratio  $\sigma$ ) if  $\text{opt}(G)/\mathcal{A}(G) \leq \sigma$  for every input graph  $G$ , where  $\mathcal{A}(G)$  is the objective value output by  $\mathcal{A}$  when applied to  $G$ . It is known that the maximum clique size cannot be approximated in polynomial time within a factor of  $n^{1-\varepsilon}$  for any  $\varepsilon > 0$  unless  $P = NP$  [5, 6, 67]. Since clique is a special case of  $s$ -clique and  $s$ -club, where  $s = 1$ , all these results apply to the versions of the maximum  $s$ -clique and maximum  $s$ -club problems that allow for *arbitrary* (non-fixed, instance-dependent)  $s$ . However, these results do not directly extend to the maximum  $s$ -clique and maximum  $s$ -club problems for the fixed constant parameter  $s > 1$ , which is given as a part of the problem definition rather than as an instance-dependent parameter. Therefore, in recent years there has been a considerable amount of research towards characterizing these problems in terms of their computational complexity in general and restricted graph classes.

Bourjolly et al. [17] use a reduction from CLIQUE to show that the maximum  $s$ -club problem is NP-hard for any fixed  $s$ . Balasundaram et al. [10] use an alternative reduction from CLIQUE to prove that both the maximum  $s$ -clique and maximum  $s$ -club problem are NP-hard, even if restricted to graphs of fixed diameter  $s + 1$ . Note that both problems are trivial when the graph's diameter is bounded above by  $s$ , therefore the transition in complexity is sudden.

Asahiro et al. [7] proved that for any  $\varepsilon > 0$  and a fixed  $s \geq 2$  the maximum  $s$ -club problem is NP-hard to approximate within a factor of  $n^{1/2-\varepsilon}$  in general graphs, improving on the hardness of  $n^{1/3-\varepsilon}$ -approximation result of Marínček and Mohar [45]. They also designed a simple polynomial-time algorithm that approximates the maximum  $s$ -club within a factor of  $n^{1/2}$  for an even  $s$ , and within a factor of  $n^{2/3}$  for an odd  $s$ . Given a graph  $G = (V, E)$ , the algorithm finds a maximum degree vertex in the power- $\lfloor s/2 \rfloor$  graph  $G^{\lfloor s/2 \rfloor} = (V, E^{\lfloor s/2 \rfloor})$  and outputs its closed

neighborhood in  $G^{\lfloor s/2 \rfloor}$ , which forms an  $s$ -club  $C_s$  of size  $\Delta(G^{\lfloor s/2 \rfloor}) + 1$  in  $G$ . To establish the approximation ratio, they consider two cases,  $\Delta(G) \geq n^{1/s}$  and  $\Delta(G) < n^{1/s}$ . Then in the first case we have:

$$\frac{\bar{w}_s(G)}{|C_s|} = \frac{\bar{w}_s(G)}{\Delta(G^{\lfloor s/2 \rfloor}) + 1} \leq \frac{\bar{w}_s(G)}{\Delta(G) + 1} < n^{1-1/s}.$$

In the second case, noting that  $\bar{w}_s(G) \leq 1 + \Delta(G) + \Delta(G)^2 + \dots + \Delta(G)^s$ , the following holds:

$$\frac{\bar{w}_s(G)}{|C_s|} \leq \frac{\Delta(G)^s + O(\Delta(G)^{s-1})}{\Delta(G) + 1} = O(\Delta(G)^{s-1}) = O(n^{1-1/s}).$$

Thus, in both cases the approximation ratio of the algorithm is  $O(n^{1-1/s})$ , which becomes  $O(n^{1/2})$  for  $s = 2$  and  $O(n^{2/3})$  for  $s = 3$ . To show that the algorithm is, in fact  $O(n^{1/2})$ -approximate for any even  $s \geq 4$ , observe that

$$\bar{w}_s(G) \leq \bar{w}_2(G^{s/2}), \quad (1)$$

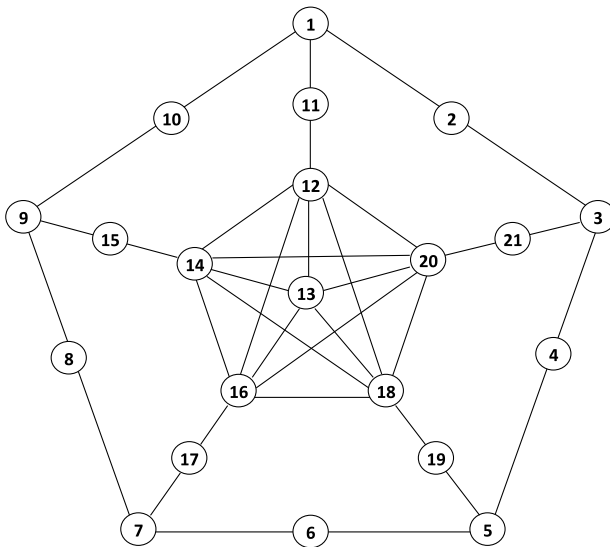
while the output of the approximation algorithm applied to the maximum  $s$ -club problem on  $G$  and to the maximum 2-club problem on  $G^{s/2}$  is the same. Thus, the approximation ratio of  $O(n^{1/2})$  holds for any even  $s$ .

It should be noted that [7] uses a stronger claim,  $\bar{w}_s(G) = \bar{w}_2(G^{s/2})$  instead of (1) in the proof of the approximation ratio. However, the equality does not hold in general, that is, we may have  $\bar{w}_s(G) < \bar{w}_2(G^{s/2})$  as in the graph in Fig. 2. The proof still holds using the inequality (1) instead.

In addition to the above results, Asahiro et al. [7] proved that for any  $\varepsilon > 0$  the maximum  $s$ -club problem is NP-hard to approximate within a factor of  $n^{1/3-\varepsilon}$  for chordal and split graphs with even  $s$ , for bipartite graphs with  $s \geq 3$ , and for  $k$ -partite graphs ( $k \geq 3$ ) with  $s \geq 2$ . On the other hand, the problem can be solved in polynomial time for chordal and split graphs with odd  $s$ , as well as for trees and interval graphs [7, 58]. Unlike the maximum  $s$ -club problem with  $s \geq 3$ , the maximum 2-club problem can be solved in  $O(n^5)$  on bipartite graphs [58]. In addition, the maximum 2-club can be approximated within a factor of  $n^{1/3}$  for split graphs.

In several recent papers, the maximum  $s$ -club problem was approached from the *parameterized complexity* perspective [23, 36, 37, 59]. In this framework, one considers a parameter  $k$  (such as the size of a structure sought) in addition to the traditional input size  $n$  [31]. A parameterized problem is *fixed-parameter tractable* if there exists an algorithm (referred to as an fpt-algorithm) that solves the parameterized problem in time  $f(k) \cdot n^{O(1)}$ , where  $f$  is a computable (typically exponential) function that depends only on the parameter  $k$ .

It is known that deciding if a given graph contains a clique of size  $k$  is W[1]-complete [24], meaning that an fpt-algorithm is unlikely to exist. In contrast, Chang et al. [23] have shown that the problem of deciding if a given graph contains an  $s$ -club of size  $k$  is fixed-parameter tractable for  $s > 1$ . The proof is as follows. Let  $G = (V, E)$  be the given graph. If  $G^{\lfloor s \rfloor}$  has a vertex  $v$  such that  $|N_{G^{\lfloor s/2 \rfloor}}[v]| \geq k$  then



**Fig. 2** A graph with  $\overline{\omega}_s(G) = 14$  (a maximum 2-club is given by, e.g.,  $C = \{5, 6, 7, 8, 9, 12, 13, 14, 16, 17, 18, 19, 20, 21\}$ ) and  $\overline{\omega}_2(G^{s/2}) = 16$  (all vertices excluding 2, 4, 6, 8, 10 form the maximum 2-club in  $G^2$ ), where  $s = 4$

$N_{G^{\lfloor s/2 \rfloor}}[v]$  is an  $s$ -club of size at least  $k$  in  $G$ . Otherwise,  $|N_{G^{\lfloor s/2 \rfloor}}[v]| < k$  for any  $v \in V$ , and it can be shown that  $|N_{G^s}[v]| < k^2$  when  $s$  is even and  $|N_{G^s}[v]| < k^3$  when  $s$  is odd [23]. Since any  $s$ -club  $C$  is a subset of  $N_{G^s}[v]$  for any  $v \in C$ , in order to check whether  $G$  has an  $s$ -club of size  $k$  it suffices to check all  $k$ -element subsets of  $N_{G^s}[v]$  for each  $v \in V$ . There are at most  $\binom{k^3}{k}n$  such subsets, and checking whether a  $k$ -element vertex set forms an  $s$ -club can be done in  $k^3$  time. Thus, the overall run time is  $O(k^{3(k+1)}n)$ .

Schäfer et al. [59] have shown that the maximum  $s$ -club problem is fixed-parameter tractable not only with the solution size  $k$  used as the parameter, but also when parameterized by the so-called dual parameter  $d = |V| - k$ . The algorithm they propose for this case runs in  $O(2^d nm)$ , where  $m$  is the number of edges in the graph. These ideas are extended to develop a practical algorithm for 2-club in [36], as will be discussed in more detail in Sect. 6. In addition, Hartung et al. [36] analyzed parameterized complexity of  $s$ -club with other parameters, such as the size of a vertex cover, feedback edge set size, size of a cluster editing set, and treewidth of the graph.

## 4 Mathematical Programming Formulations

In this section, mathematical programming formulations for the maximum  $s$ -clique and maximum  $s$ -club problems are presented. The maximum clique problem is one

of the well studied problems in discrete optimization, with a number of known integer, as well as continuous non-convex formulations [15]. Similar formulations can be applied to the maximum  $s$ -clique problem on a graph  $G$  by reducing it to the maximum clique problem on the  $s$ th power of  $G$ ,  $G^s = (V, E^s)$ , constructed from the original graph as mentioned above. Let  $\overline{E^s}$  denote the complement set of edges in  $G^s$ , that is,  $\overline{E^s} = \{(i, j) : i, j \in V, i < j, d_G(i, j) > s\}$ . Then the following formulation of the maximum clique problem written for the power- $s$  graph  $G^s$  can be used for the maximum  $s$ -clique problem on  $G$ :

$$\text{Maximize (max)} \quad \sum_{i \in V} x_i \quad (2)$$

$$\text{subject to (s.t.):} \quad x_i + x_j \leq 1 \quad \forall (i, j) \in \overline{E^s}, \quad (3)$$

$$x_i \in \{0, 1\} \quad \forall i \in V. \quad (4)$$

The first mathematical program for computing the  $s$ -club number of a graph was proposed in [17]; see also [10]. In the following, we explain this general integer programming model first and then describe special cases for  $s = 2, 3$  that are of highest practical interest and have received more attention in the literature. For  $S \subseteq V$  the vector  $x \in \{0, 1\}^n$  such that  $x_i = 1$  if and only if  $i \in S$  is called the *characteristic vector* of  $S$ . In the general model (5)–(8) below, which is often referred to as *chain formulation*, the vector of decision variables  $x$  is the characteristic vector of the  $s$ -club sought. For every pair of vertices  $i, j \in V$ , let  $P_{ij}^s$  be the set of all paths of length at most  $s$  between  $i$  and  $j$  in  $G$ . We will denote by  $\mathbb{P}$  the set of all such paths in  $G$ , i.e.,  $\mathbb{P} = \bigcup_{i, j \in V} P_{ij}^s$ . Let  $V_P$  be the set of vertices included in a path  $P$ . Also let  $y_P$  be the auxiliary binary variable associated with every path  $P \in \mathbb{P}$ . If this variable is equal to 1 in a feasible solution, this implies that all the vertices in the path  $P$  are included in the corresponding  $s$ -club. Then the following finds the maximum cardinality  $s$ -club in  $G$ :

$$\max \quad \sum_{i \in V} x_i \quad (5)$$

$$\text{s.t.:} \quad x_i + x_j \leq 1 + \sum_{P \in P_{ij}^s} y_P \quad \forall (i, j) \notin E, \quad (6)$$

$$y_P \leq x_i \quad \forall P \in \mathbb{P}, \forall i \in V_P, \quad (7)$$

$$x_i, y_P \in \{0, 1\} \quad \forall i \in V, \forall P \in \mathbb{P}. \quad (8)$$

In this formulation, constraint (6) ensures that two vertices  $i$  and  $j$  such that  $d_G(i, j) > s$  cannot both belong to the same  $s$ -club (in this case  $P_{ij}^s = \emptyset$  and the constraint becomes  $x_i + x_j \leq 1$ ). It also guarantees that if two nonadjacent vertices are included in the  $s$ -club sought, then there must be at least one path of length at most  $s$  such that all the vertices from this path are also included in the  $s$ -club. In addition, constraint (7) forces  $y_P$  to be 0 whenever a vertex from  $P$  is not included

in the  $s$ -club. For  $s = 2$  the above chain formulation becomes:

$$\max \sum_{i \in V} x_i \quad (9)$$

$$\text{s.t.: } x_i + x_j \leq 1 + \sum_{k \in N_G(i) \cap N_G(j)} x_k \quad \forall (i, j) \notin E, \quad (10)$$

$$x_i \in \{0, 1\} \quad \forall i \in V. \quad (11)$$

This model ensures that any two nonadjacent vertices that are in the same 2-club must have at least one common neighbor inside the 2-club.

Considering the number of possible distinct paths of length at most  $s$  between every pair of vertices, the chain formulation may have an excessive number of variables when  $s > 2$ . In general, we may have  $|P_{ij}^s| = O(n^{s-1})$  for every pair of vertices, so  $|\mathbb{P}| = O(n^{s+1})$ . Therefore, this model does not scale well when  $s$  increases, and even solving small instances using this formulation is challenging when  $s \geq 3$  [65].

To formulate the maximum 3-club problem using a smaller number of variables, the *neighborhood formulation* (13)–(16) was proposed in [4] that has  $|V| + |E|$  variables. Note that a pair of nonadjacent vertices  $i$  and  $j$  in  $G$  can be a part of the same 3-club  $S$  only if they have a common neighbor  $k$  in  $S$  or there are two adjacent vertices  $\{p, q\} \in S$  such that  $p \in N_G(i)$  and  $q \in N_G(j)$ . The first condition holds if and only if  $d_{G[S]}(i, j) = 2$ . If the first condition does not hold and the second condition holds, then  $p \in \{N_G(i) \setminus N_G(j)\}$  and  $q \in \{N_G(j) \setminus N_G(i)\}$ . Let  $E_{ij}$  denote the set of edges that connect such intermediate nodes for  $i$  and  $j$ :

$$E_{ij} = \{(p, q) \in E : p \in \{N_G(i) \setminus N_G(j)\}, q \in \{N_G(j) \setminus N_G(i)\}\} \\ \forall i, j : d_G(i, j) = 3. \quad (12)$$

Now associate a binary variable  $x_i$  with each vertex  $i \in V$  and a binary variable  $z_{ij}$  with each edge  $(i, j) \in E$ . Then the maximum 3-club problem in  $G = (V, E)$  can be formulated using the following binary program:

$$\max \sum_{i \in V} x_i \quad (13)$$

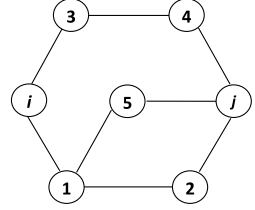
$$\text{s.t.: } x_i + x_j \leq 1 + \sum_{k \in N_G(i) \cap N_G(j)} x_k + \sum_{(p, q) \in E_{ij}} z_{pq} \quad \forall (i, j) \notin E, \quad (14)$$

$$z_{ij} \leq x_i, \quad z_{ij} \leq x_j, \quad z_{ij} \geq x_i + x_j - 1 \quad \forall (i, j) \in E, \quad (15)$$

$$x_i, z_{ij} \in \{0, 1\} \quad \forall i \in V, \forall (i, j) \in E. \quad (16)$$

Neighborhood constraints (14) ensure that two nonadjacent vertices  $i$  and  $j$  cannot be both in the solution unless their common neighbor is in the solution or a pair of their neighbors  $p$  and  $q$ , linked by an edge, are in the solution. The constraints

**Fig. 3** Subgraph  $G_{ij}$  illustrating the node cut set formulation



in (15) guarantee that an edge  $(i, j)$  is used if and only if both its endpoints belong to the solution. The neighborhood formulation has  $|V| + |E|$  variables and  $\frac{|V|^2 - |V|}{2} + 2|E|$  constraints.

Almeida and Carvalho [4] also proposed another formulation for the maximum 3-club problem that is based on identifying minimal node cut sets for every pair of vertices with  $d_G(i, j) = 3$ . Consider a pair of nonadjacent vertices  $i, j \in G$  and let  $E_{ij}$  be defined as in (12). Recall that  $E_{ij}$  is the set of inner edges of chains with length 3 connecting  $i$  and  $j$  with no common neighbors. Let  $V_{ij}$  represent the set of vertices incident to edges from  $E_{ij}$ . We associate with  $i$  and  $j$  a subgraph  $G_{ij} = (V'_{ij}, E'_{ij})$  where  $V'_{ij} = V_{ij} \cup \{i, j\}$  and  $E'_{ij} = E_{ij} \cup \{(i, v) \in E : v \in V_{ij}\} \cup \{(j, v) \in E : v \in V_{ij}\}$ . Figure 3 is an example of a subgraph  $G_{ij}$  in which  $E_{ij} = \{(1, 2), (1, 5), (3, 4)\}$ ,  $V_{ij} = \{1, 2, 3, 4, 5\}$  and  $E'_{ij} = \{(1, 2), (1, 5), (3, 4), (i, 1), (i, 3), (2, j), (4, j), (5, j)\}$ . Let  $S_{ij}$  be an  $i$ - $j$  node cut set and define  $\mathcal{S}_{ij}^M$  to be the set of all minimal  $S_{ij}$ . For our example in the figure,  $\mathcal{S}_{ij}^M = \{\{1, 3\}, \{1, 4\}, \{2, 3, 5\}, \{2, 4, 5\}\}$ . These sets separate  $i$  and  $j$  in  $G_{ij}$ , therefore, to include nodes  $i$  and  $j$  with  $d_G(i, j) = 3$  in the same 3-club  $S$ , it is necessary to include a node of each set  $S_{ij} \in \mathcal{S}_{ij}^M$ . Thus, the *node cut set formulation* (17)–(19) for the maximum 3-club problem can be stated as follows:

$$\max \sum_{i \in V} x_i \quad (17)$$

$$\text{s.t.: } x_i + x_j \leq 1 + \sum_{k \in N_G(i) \cap N_G(j)} x_k + \sum_{s \in S_{ij}} x_s \quad \forall (i, j) \notin E, S_{ij} \in \mathcal{S}_{ij}^M, \quad (18)$$

$$x_i \in \{0, 1\} \quad \forall i \in V. \quad (19)$$

In this formulation, inequalities (18) are the node cut set constraints described above. The formulation has  $|V|$  variables, but potentially exponential number of constraints. Note that constraints associated with non-minimal cut sets are dominated by constraints (18) and are not necessary.

Next, we present an integer programming formulation for the maximum  $s$ -club problem recently proposed by Veremyev and Boginski [65]. We first discuss the formulation for  $s = 2$  and then extend it to the higher  $s$  values. Let  $V = \{1, \dots, n\}$  and let  $A = [a_{ij}]_{i,j=1}^n$  be the adjacency matrix of  $G = (V, E)$ . Then the characteristic

vector  $x$  of a 2-club  $S$  must satisfy the following nonlinear constraint:

$$a_{ij} + \sum_{k \in V} a_{ik} a_{kj} x_k \geq x_i x_j \quad \forall i, j \in V. \quad (20)$$

Linearizing this constraint, we formulate the maximum 2-club problem as follows:

$$\max \quad \sum_{i \in V} x_i \quad (21)$$

$$\text{s.t.:} \quad a_{ij} + \sum_{k=1}^n a_{ik} a_{kj} x_k \geq x_i + x_j - 1 \quad \forall i, j \in V, \quad (22)$$

$$x_i \in \{0, 1\} \quad \forall i \in V. \quad (23)$$

The above formulation can be simplified as follows:

$$\max \quad \sum_{i \in V} x_i \quad (24)$$

$$\text{s.t.:} \quad \sum_{k \in N_G(i) \cap N_G(j)} x_k \geq x_i + x_j - 1 \quad \forall (i, j) \notin E \quad (25)$$

$$x_i \in \{0, 1\} \quad \forall i \in V. \quad (26)$$

Similarly, the characteristic vector  $x$  of a 3-club must satisfy the following nonlinear constraints:

$$\sum_{k=1}^n a_{ik} a_{kj} x_k + \sum_{k=1}^n \sum_{m=1}^n a_{ik} a_{km} a_{mj} x_k x_m \geq x_i x_j \quad \forall (i, j) \notin E. \quad (27)$$

Letting  $w_{ij} = x_i x_j$  for all  $i, j \in V$  and linearizing the constraints, we obtain the following formulation for the maximum 3-club problem:

$$\max \quad \sum_{i=1}^n x_i \quad (28)$$

$$\text{s.t.:} \quad \sum_{k=1}^n a_{ik} a_{kj} x_k + \sum_{k=1}^n \sum_{m=1}^n a_{ik} a_{km} a_{mj} w_{km} \geq x_i + x_j - 1 \quad \forall (i, j) \notin E, \quad (29)$$

$$w_{ij} \leq x_i, w_{ij} \leq x_j, w_{ij} \geq x_i + x_j - 1 \quad \forall i, j \in V, \quad (30)$$

$$x_i, w_{ij} \in \{0, 1\} \quad \forall i, j \in V. \quad (31)$$

This formulation contains  $O(n^2)$  binary variables and  $O(n^2)$  constraints. Similarly, one can develop the model and linearize it using the standard approaches for the general case of the maximum  $s$ -club problem. However, the resulting formulation will have  $O(n^{s-1})$  variables. Veremyev and Boginski [65] exploit the special structure



of  $s$ -club and propose an efficient linearization technique that reduces the number of variables substantially. We discuss their compact binary formulation next.

Consider a subset of vertices  $S$  and its characteristic vector  $x$ . Let  $v_{ij}^{(l)}$  ( $i, j = 1, \dots, n; l = 2, \dots, s$ ) be a binary variable taking the value 1 if there exists at least one path of length  $l$  from  $i$  to  $j$  in  $G[S]$  and 0 otherwise. Note that for  $l = 2$  we have  $v_{ij}^{(2)} = \min\{x_i x_j \sum_{k=1}^n a_{ik} a_{kj} x_k, 1\}$ , which can be linearized using the following set of constraints:

$$\begin{cases} v_{ij}^{(2)} \leq x_i, & v_{ij}^{(2)} \leq x_j \\ v_{ij}^{(2)} \leq \sum_{k=1}^n a_{ik} a_{kj} x_k, & v_{ij}^{(2)} \geq \frac{1}{n} \left( \sum_{k=1}^n a_{ik} a_{kj} x_k \right) + (x_i + x_j - 2). \end{cases}$$

Other variables for higher values of  $l = 3, \dots, s$  can be found recursively using  $v_{ij}^{(l)} = \min\{x_i \sum_{k=1}^n v_{kj}^{(l-1)} a_{ik}, 1\}$  and linearized by applying the following set of inequalities:

$$\begin{cases} v_{ij}^{(l)} \leq x_i, & v_{ij}^{(l)} \leq \sum_{k=1}^n a_{ik} v_{kj}^{(l-1)} \\ v_{ij}^{(l)} \geq \frac{1}{n} \left( \sum_{k=1}^n a_{ik} v_{kj}^{(l-1)} \right) + (x_i - 1). \end{cases}$$

Therefore the maximum  $s$ -club problem can be formulated as the following binary linear program:

$$\max \quad \sum_{i=1}^n x_i \quad (32)$$

s.t.:

$$\sum_{l=2}^k v_{ij}^{(l)} \geq x_i + x_j - 1 \quad \forall (i, j) \notin E, \quad (33)$$

$$v_{ij}^{(2)} \leq x_i, \quad v_{ij}^{(2)} \leq x_j, \quad v_{ij}^{(2)} \leq \sum_{k=1}^n a_{ik} a_{kj} x_k \quad \forall i, j \in V, i < j, \quad (34)$$

$$v_{ij}^{(2)} \geq \frac{1}{n} \left( \sum_{k=1}^n a_{ik} a_{kj} x_k \right) + (x_i + x_j - 2) \quad \forall i, j \in V, i < j, \quad (35)$$

$$v_{ij}^{(l)} \leq x_i, \quad v_{ij}^{(l)} \leq \sum_{k=1}^n a_{ik} v_{kj}^{(l-1)} \quad \forall i, j \in V, i < j, l = 3, \dots, s, \quad (36)$$

$$v_{ij}^{(l)} \geq \frac{1}{n} \left( \sum_{k=1}^n a_{ik} v_{kj}^{(l-1)} \right) + (x_i - 1) \quad \forall i, j \in V, i < j, l = 3, \dots, s, \quad (37)$$

$$x_i, v_{ij}^{(l)} \in \{0, 1\} \quad \forall i, j \in V, i < j, l = 2, \dots, s. \quad (38)$$

The above model is the most compact known formulation for the maximum  $s$ -club problem with  $O(sn^2)$  variables and constraints. For more information about compact formulation and its properties, we refer the reader to [65].

## 5 Polyhedral Results

Due to the structure of the problem and its dependence on the value of parameter  $s$ , most of the research in this area has been focused on the 2-club polytope and, partially, 3-club polytope and not on the  $s$ -club polytope in general. In this section, we review the polyhedral results available for the 2-club polytope.

Consider a nontrivial simple undirected connected graph  $G = (V, E)$ . A subset of vertices  $I \subseteq V$  is a *2-independent set* in  $G$  if  $d_G(i, j) > 2 \forall i, j \in I$ . Let  $M_1$  be the edge-vertex incidence matrix of the complement graph  $\bar{G}$ . The rows of  $M_1$  correspond to edges  $(i, j) \in \bar{E}$  and the columns correspond to vertices  $i \in V$ . The entries in the row corresponding to an edge  $(i, j)$  are 1 in columns  $i$  and  $j$  and are 0 otherwise. Let  $M_2$  be the matrix representing the common neighborhood of  $i, j$  for every  $(i, j) \in \bar{E}$ . The rows of  $M_2$  correspond to edges  $(i, j) \in \bar{E}$  and the columns correspond to vertices  $i \in V$ . The entries in the row corresponding to an edge  $(i, j)$  are 1 in columns  $k \in N_G(i) \cap N_G(j)$  and are 0 otherwise. Let  $A = M_1 - M_2$ , then the maximum 2-club problem formulation (9)–(11) can be written as [10]:

$$\bar{\omega}_2(G) = \max \{ \mathbf{1}^T x : Ax \leq \mathbf{1}, x \in \{0, 1\}^{|V|} \},$$

where  $\mathbf{1}$  is the vector of all ones of appropriate dimension and  $\bar{\omega}_2(G)$  is the 2-club number of  $G$ . Let  $Q$  be the set of feasible binary vectors defined as  $Q = \{x \in \{0, 1\}^{|V|} : Ax \leq \mathbf{1}\}$ , then the 2-club polytope  $P_{2C}$  is given by the convex hull of  $Q$ :  $P_{2C} = \text{conv}(Q)$ . The following results were established in [10].

1.  $\dim(P_{2C}) = |V|$ .
2.  $x_i \geq 0$  induces a facet of  $P_{2C}$  for every  $i \in V$ .
3. For any  $i \in V$ ,  $x_i \leq 1$  induces a facet of  $P_{2C}$  if and only if  $d_G(i, j) \leq 2 \forall j \in V$ .
4. Let  $I$  be a maximal 2-independent set in  $G$ . Then  $\sum_{i \in I} x_i \leq 1$  induces a facet of  $P_{2C}$ .

Note that each neighborhood constraint is associated with two vertices  $v$  and  $v'$  such that  $d_G(v, v') = 2$ . For any node  $i \in V \setminus \{v, v'\}$  such that  $\min\{d_G(i, v), d_G(i, v')\} > 2$ , the inequality  $x_v + x_{v'} + x_i - \sum_{j \in \{N_G(v) \cap N_G(v')\}} x_j \leq 1$  is valid for  $P_{2C}$  because neither  $v$  nor  $v'$  can be included in a 2-club that includes node  $i$ , and to include nodes  $v$  and  $v'$ , at least one of their common neighbors must also be included in the 2-club. This inequality is a lifted version of the neighborhood constraint (10). Carvalho and

Almeida [22] used this observation to establish the following valid inequality for  $P_{2C}$ :

$$\sum_{i \in I \cup \{v, v'\}} x_i - \sum_{j \in N_G(v) \cap N_G(v')} x_j \leq 1, \quad (39)$$

where  $I \subseteq V \setminus \{v, v'\}$  is such that  $I \cup \{v\}$  and  $I \cup \{v'\}$  are 2-independent sets in  $G$ . They also extended this result to triples of vertices as follows. Let  $v, v', v''$  be given. For a vertex  $j$  denote by  $a_j = (|N_G(j) \cap \{v, v', v''\}| - 1)^+$ , where  $a^+ = \max\{a, 0\}$ , i.e.,  $a_j = 2$  if  $j$  neighbors all three vertices;  $a_j = 1$  if  $j$  neighbors two of the three vertices; and  $a_j = 0$ , otherwise. Let  $R = \{v, v', v''\}$  be an independent set in  $G$ . Let  $I \subseteq V \setminus \{v, v', v''\}$  be such that  $I \cup \{v\}$ ,  $I \cup \{v'\}$  and  $I \cup \{v''\}$  are 2-independent sets in  $G$ . Then the inequality

$$\sum_{i \in I \cup \{v, v', v''\}} x_i - \sum_{j \in V} (|N_G(j) \cap \{v, v', v''\}| - 1)^+ x_j \leq 1 \quad (40)$$

is valid for  $P_{2C}$ .

More recently, Mahdavi [43] developed a family of valid inequalities that subsume both (39) and (40).

**Theorem 1** ([43]) *Let  $I$  be an independent set in  $G$ . Then the inequality*

$$\sum_{i \in I} x_i - \sum_{j \in V \setminus I} (|N_G(j) \cap I| - 1)^+ x_j \leq 1 \quad (41)$$

*is valid for  $P_{2C}$ . If, in addition,  $I$  is a distance 2-dominating set, i.e.,  $I$  is an independent distance 2-dominating set (I2DS), then this inequality defines a facet for  $P_{2C}$  referred to as I2DS facet.*

Mahdavi [43] also proved that given a noninteger feasible point  $\tilde{x}$  in  $P_{2C}$  deciding whether this point violates an I2DS inequality is NP-complete, that is, the I2DS inequalities separation problem is NP-complete. On a positive note, I2DS inequalities are sufficient to derive the complete description of the 2-club polytope of trees. See [43] for a more detailed discussion on the 2-club polytope.

As for the maximum 3-club problem, Almeida and Carvalho [4] developed some non-trivial valid inequalities based on ideas similar to those used to develop (39) and (40) above. An interesting future research question is whether the valid inequalities they developed for 3-club can be generalized to develop a class of facets similar to I2DS above.

## 6 Exact and Heuristic Algorithms

The correspondence between the maximum clique and maximum  $s$ -clique problems implies that the heuristic and exact algorithms for maximum clique problem can be

applied to the  $s$ th power of the graph to solve the maximum  $s$ -clique problem. In such cases, the performance of these algorithms may be poor as the edge density is higher in  $G^s$ . Unlike the maximum clique problem, the maximum  $s$ -clique problem has not been the subject of extensive research and we are not aware of any computational results for this problem to date. This may be due to the above-mentioned correspondence between the two problems which facilitates the use of proposed algorithms for clique detection to solve the later case. Therefore, in this section our focus will primarily be on the existing algorithms for the maximum  $s$ -club problem.

Due to the NP-hardness of checking whether an  $s$ -club is maximal, developing sufficient conditions for an  $s$ -club to be maximal is of importance for designing effective algorithmic procedures for the maximum  $s$ -club problem. One such condition, which was proposed in [63], is outlined next.

For a positive integer  $k$ , the  $k$ -neighborhood  $N_G^k(v)$  of  $v \in V$  is given by  $N_G^k(v) = \{v' : d_G(v, v') \leq k\}$ . Note that  $v \in N_G^k(v)$  for any  $k \geq 1$ , thus  $N_G^1(v) = N_G[v]$ . The  $k$ -neighborhood  $N_G^k(S)$  of a given subset of vertices  $S$  is defined as follows:  $N_G^k(S) = \bigcap_{v \in S} N_G^k(v)$ . Given an  $s$ -club  $S$  and a positive integer  $p$ , we recursively define  $N^s(G, S, p)$  as the  $s$ -neighborhood of  $S$  in  $G[N^s(G, S, p - 1)]$ :

$$N^s(G, S, p) = \begin{cases} N_G^s(S), & \text{if } p = 1, \\ N^s(G[N^s(G, S, p - 1)], S, 1), & \text{if } p \geq 2. \end{cases}$$

Then the following properties hold [63]:

1. If  $S \subseteq S^* \subseteq V$ , then we have  $N^s(G[S], S, p) \subseteq N^s(G[S^*], S, p) \subseteq N^s(G, S, p)$  for any  $p \geq 1$ .
2. If  $S$  is an  $s$ -club, then we have  $S = N^s(G[S], S, p) \subseteq N^s(G, S, p + 1) \subseteq N^s(G, S, p)$  for any  $p \geq 1$ .
3. Let  $S$  be an  $s$ -club that is not maximal. Then for any maximal  $s$ -club  $S^*$  containing  $S$  we have:  $S^* \subseteq N^s(G, S, p)$  for any  $p \geq 1$ .

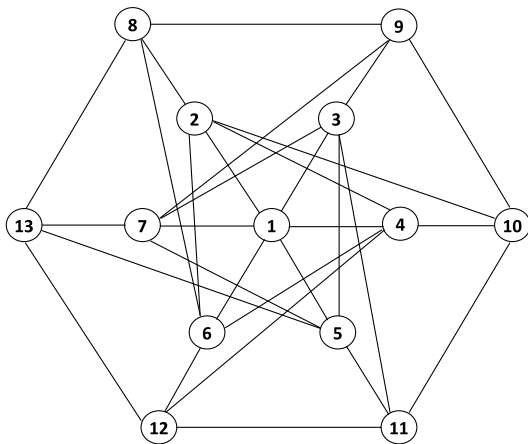
**Theorem 2 ([63])** *Let  $S$  be an  $s$ -club. If there exists a positive integer  $p$  such that  $N^s(G, S, p) = S$ , then  $S$  is a maximal  $s$ -club.*

The sufficient condition in Theorem 2 is not necessarily satisfied by a maximal  $s$ -club, as illustrated by the example in Fig. 4. In the graph  $G = (V, E)$  in this figure, the subset of vertices  $S = \{1, 2, 3, 4, 5, 6, 7\}$  is a maximal 2-club, however, for any  $p > 1$ ,  $N^2(G, S, p) = N^2(G, S, 1) = V \supset S$ . Since the distance between vertices 8 and 11 is 3,  $V$  is not a 2-club.

## 6.1 Heuristic Algorithms

Due to computational intractability of the maximum  $s$ -club problem, heuristics become the method of choice for solving the maximum  $s$ -club problem in practice.

**Fig. 4** A graph with a maximal  $s$ -club  $S = \{1, 2, 3, 4, 5, 6, 7\}$  that does not satisfy the condition of Theorem 2



Several construction heuristics have been proposed in the literature. In 2000, Bourjolly et al. [16] proposed three simple heuristic algorithms for the maximum  $s$ -club problem called DROP, CONSTELLATION and  $s$ -CLIQUE-DROP. Among these, DROP, which runs in  $O(|V|^3|E|)$  time, was reported to produce the best result in terms of solution quality specially in graphs with higher density. DROP works as follows: we start with the whole graph  $G$  and at each iteration the vertex  $i$  with most infeasibility is deleted where infeasibility is defined as the number  $q_i$  of vertices of  $G$  whose shortest distance to  $i$  has a length of at least  $s + 1$ . If there is a tie, a vertex of minimum degree is then selected for elimination and the graph is updated. The procedure continues until no infeasible vertex can be found. CONSTELLATION is based on identifying the largest star graph in the first step. In the next iteration, the vertex having the largest number of neighbors in the remaining graph is selected and added to the  $s$ -club provided that the total number of iterations does not exceed  $s - 1$ . CONSTELLATION runs in  $O(s(|V| + |E|))$  time and was reported to perform well solving the maximum 2-club problem on low density graphs. The third algorithm,  $s$ -CLIQUE-DROP, proceeds by identifying the largest  $s$ -clique in  $G$  and removing all vertices not belonging to  $s$ -clique from  $G$  along with their incident edges. Then DROP is called to find a feasible solution. To obtain the largest  $s$ -clique, the maximum clique problem is solved on  $G^s$  using one of the existing algorithms. The recently proposed EXPAND algorithm [63] proceeds by taking the closed neighborhood of each vertex  $i$  in  $G$ , computing its  $s$ -neighborhood and applying DROP procedure on the subgraph induced by  $s$ -neighborhood of a vertex to obtain a feasible solution. Recall that the closed neighborhood of a vertex provides a 2-club, which is used to obtain the  $s$ -neighborhood for  $s \geq 2$ . To increase the chance of obtaining larger  $s$ -clubs in the initial steps of EXPAND, vertices are sorted based on a non-increasing order of their degrees in the initialization step. Therefore at each iteration  $i$ , DROP is called only if the size of the  $s$ -neighborhood of vertex  $i$  is larger than the best  $s$ -club found so far. A similar version of EXPAND, named IDROP, was independently used in [23].

Recently, a variable neighborhood search (VNS) meta-heuristic has been proposed for the maximum  $s$ -club problem [63]. VNS is based on the idea that a local optimum for one neighborhood structure is not necessarily a local optimum for another neighborhood structure [48]. Hence, to avoid being trapped in a poor-quality local optimum, the VNS explores, in a systematic fashion, multiple neighborhood structures. As a part of the procedure, the sufficient condition described in Theorem 2, has been used to test the maximality of a given solution to search the solution space more effectively. Computational experiments for  $s = 2, 3$  using VNS algorithm were conducted on two sets of instances. The first testbed included the same set of random instances used in [44] to facilitate the comparison between the two methods. The second set included some of the large-scale instances from the tenth DIMACS implementation challenge [30]. For more detail on the proposed algorithm, neighborhood structures and computational results see [63].

## 6.2 Exact Algorithms

The first exact algorithm for the maximum  $s$ -club problem was proposed by Bourjolly et al. [17]. The proposed branch-and-bound (B&B) algorithm employs DROP heuristic to direct its branching process. For the bounding process, algorithm relies on the solutions to the maximum stable set problem solved on an auxiliary graph. Two branches are generated at the root node of the search tree that correspond to removing or keeping the vertex selected by a single iteration of DROP. The algorithm first explores the branch that removes the vertex. The process is then recursively applied until a terminal node is reached, yielding a depth first search. Note that deciding to remove or keep a vertex during the branching process may increase the shortest chains length and this affects the whole subtree rooted at the node in which this decision has been made. As a result, a pair of vertices in the current solution at some node of the subtree may appear too far away from each other. This leads to an infeasible solution and thus the corresponding branch is pruned. For the upper bounding procedure, let  $G' = (V', E')$  be the graph induced by the current solution at a given node of the B&B tree and let  $H = (V', F)$  be an auxiliary graph associated with  $G'$ , where there is an edge between any two vertices in  $H$  only if the shortest path connecting these two vertices in  $G'$  has length greater than  $s$ . Obviously, if there is an edge between two vertices in  $H$ , they cannot both belong to the same  $s$ -club in  $G'$ . Therefore, the largest independent set in  $H$  provides an upper bound on the size of the largest  $s$ -club in  $G'$ . In their computational results, authors report the average solution size and CPU time for  $s = 2, 3, 4$  on randomly generated instances with different edge densities. Instances were generated using the method proposed in [33].

Recently, Chang et al. [23] have shown that the B&B algorithm of Bourjolly et al. [17] runs in  $O(1.62^n)$  time and proposed a variation of this algorithm that uses IDROP procedure to find the initial feasible solution and computes the  $s$ -coloring number of the graph associated with each node of the B&B tree to obtain an upper

bound on the size of the  $s$ -club for that node. Observe that the *chromatic number*  $\chi(G)$  of  $G$  is the minimum number of colors required to color the vertices of  $G$  *properly*, that is, so that no two neighbors are assigned the same color and the  $s$ -coloring number of  $G$  is the minimum number of colors required to color all vertices such that no two vertices of distance at most  $s$  are assigned the same color. Note that the  $s$ -coloring number of  $G$  provides an upper bound on the  $s$ -club number of  $G$  and one can compute the chromatic number  $\chi(G^s)$  of the  $s$ th power graph  $G^s$  to obtain the  $s$ -coloring number of  $G$ . The authors report the results of computational experiments with a set of randomly generated instances, Erdős collaboration networks [11, 35], and some benchmark graphs from the second DIMACS implementation challenge [29].

More recently, Mahdavi and Balasundaram [44] proposed another B&B algorithm to compute the  $s$ -club number of a graph. Their algorithm employs two methods for computing a lower bound. The first method selects the larger of the two solutions found using DROP and CONSTELLATION heuristics, and the second method is a bounded enumeration-based technique. This lower-bounding scheme proceeds by finding an initial  $s$ -club  $S$  followed by a bounded search that enumerates  $s$ -clubs containing  $S$ . The idea behind this bounded search is to improve the initial solution in a reasonable amount of time. The best solution found by these two methods initializes the incumbent. Two methods are used to derive an upper bound for the B&B algorithm. The first one, proposed independently of [23], is based on obtaining the  $s$ -coloring number of graph associated with every node in the B&B tree. To obtain this upper bound a combination of greedy heuristic and DSATUR heuristic, proposed in [18], is used. The second method computes the maximum  $s$ -clique to serve as an upper bound for the  $s$ -club number of a given graph  $G$ . To obtain this upper bound, the algorithm proposed by [51] is employed to find the maximum clique on  $s$ th power graph  $G^s$ . For branching, a vertex dichotomy is used, where a vertex is selected and fixed to be included or deleted from the solution. To traverse the search tree, best bound search (BBS) strategy has been considered. Authors report extensive computational results, for  $s = 2, 3$ , using four different combinations of lower-bounding and upper-bounding techniques on a set of randomly generated instances of order  $n = 50, 100, 150$  and  $200$  with seven different densities ranging from  $0.0125$  up to  $0.25$ . They report on the effectiveness of the bounding techniques used in the B&B algorithm and their relation with the topological structure of the randomly generated instances. The general B&B framework for solving the maximum  $s$ -club problem outlined in [44] can be easily adapted to include alternative lower- and upper-bounding schemes. One such enhancement has been proposed in [63], where the B&B framework was used in conjunction with the VNS heuristic mentioned above.

Veremyev and Boginski [65] solved the maximum  $s$ -club problem for  $s = 2, \dots, 7$ , using the compact formulation (32)–(38) on a set of randomly generated instances of order  $n = 100, 200, 300$  with different edge densities. For every combination, 10 instances are generated and the average maximum  $s$ -club size, average CPU time and the average tightness for each group of problem instances have been reported. The advantage of the compact formulation is that it contains a reasonable

number of entities that grows linearly as  $s$  increases, thus providing an opportunity to solve the problem for higher values of  $s$ . Computational results show that the compact formulation is rather tight and the relative gap between the exact and the LP relaxation objective values decreases for larger values of  $s$ . The results of experiments with IP-based approaches for  $s = 2, 3$  have also been reported in [4, 22].

Hartung et al. [36] used their theoretical findings concerning parameterized algorithms for 2-clubs based on the dual parameter  $d = |V| - k$  to develop a B&B strategy in conjunction with a kernelization proposed in [59]. The results of experiments with the proposed algorithm for the maximum 2-club problem that they report are very encouraging. In particular, their implementation significantly outperforms other known exact approaches on small to medium-size random graphs and large-scale real-life networks from the tenth DIMACS implementation challenge [30].

## 7 Applications and Extensions

The introduction of the concepts of  $s$ -clique and  $s$ -club was originally motivated by applications in social networks analysis, where these distance-based clique relaxations are used to model cohesive subgroups [3, 49]. A social network can be formalized by a simple undirected graph  $G = (V, E)$ . The vertex set  $V$  can represent people, or actors, in a social network and the mutual relationships between pairs of actors can be naturally modeled using edges. For example, in a collaboration network the edges could represent collaborations between researchers. For mathematicians and computational geometers [11, 35], such collaboration networks are used to determine the collaborative distance between researchers which was first popularized by the concept of Erdős numbers [34].

Studying cohesive or “tightly knit” subgroups, which describe groups of actors that tend to share certain features of interest [60, 66], finds applications in different branches of sociology, including epidemiology of sexually transmitted diseases [55], organizational management [27], and crime detection/prevention and terrorist network analysis [12, 56, 57] among many others. For example, in [46],  $s$ -cliques and  $s$ -clubs are used to analyze 9/11 terrorist network.

Even though the distance-based clique relaxation structures may not be characterized by a very high overall degree of interaction between their members that is typical for some other models, the low distances between all group members make them appropriate models of cohesive subgroups in situations where easy reachability is most crucial. This is the case, in particular, when one deals with various types of *flows* in the network, such as flows of information, spread of diseases, or transportation of commodities. It is therefore not surprising that  $s$ -cliques and  $s$ -clubs appear naturally in many real-life complex systems, including biological and social systems, as well as telecommunication, transportation, and energy infrastructure systems.

In social networks, the proliferation of low-diameter structures manifests itself in catch-phrases “small world phenomenon” and “six degrees of separation” that made



their way to the mainstream popular culture. A low diameter is a key characteristic of many other massive-scale complex networks that tend to have *power-law* degree distribution, or the so-called *scale-free* property [50]. Such networks typically have a small number of high-degree nodes, which are most likely to be “central vertices” in the largest *s*-clubs. In biology, it has been observed that groups of proteins where interactions occur via a central protein often represent similar biological processes [8]. This phenomenon makes computing 2-clubs, especially those that induce star graphs, particularly interesting [10, 52].

In transportation, hub-and-spoke model is the most popular network architecture used by major airlines [2, 38]. One of the main advantages of this model is that it is optimal in the sense that it ensures a 2-hop reachability while using the minimum possible total number of direct connections. Under this model, most of the flights are routed through several hub airports. This provides passengers a convenient access (via hubs) to numerous destinations that would not be able to support many direct connections, as well as allows to facilitate a wide variety of services, thus attracting more customers.

Another application is in computer and communication networks security [26]. A *bot* is a malicious program carrying out tasks for other programs or users and a *botnet* is a network of *bots*. Botnets are usually controlled by members of organized crime groups, called botmasters, for many different purposes. Almost all computers can host malicious programs that belong to a particular botnet and only a few of them might be immune to becoming a host. Distribution of spam and Distributed Denial-of-Service Attacks (DDoS) are among the malicious tasks performed by botnets. Naturally, the botmaster would like to maximize the effect of attack, damage, to the network and is therefore interested in selecting the densest subnetwork to initiate the attack. Identifying the densest subnetwork would help the botmaster to pick the minimum number of nodes to attack. This strategy leads to the greatest possible damage to the network and, at the same time, minimizes the chance of detection and regulation. Therefore to protect the network and minimize the damage and, at the same time, reduce the cost of taking defensive steps, it is essential to locate cohesive subgraphs and nodes through which such malicious programs can propagate all over the network very quickly.

In internet research, 2-clubs have been used for clustering web sites to facilitate text mining in hyper-linked documents [47], as well as search and retrieval of topically related information [64]. In wireless networks, small-diameter dominating sets, or the so-called *dominating s-clubs* offer an attractive alternative to usual connected dominating sets as a tool to model virtual backbones used for routing [19, 40]. Since wireless networks are often modeled using geometric graphs known as unit disk and unit ball graphs, studying the distance-based relaxations restricted to such graphs is of special interest. It is well known that a maximum clique in unit disk graphs can be found in polynomial time [9, 25]. However, the complexity of maximum 2-clique and 2-club problems restricted to unit disk graphs remains open. A 0.5-approximation algorithm for the maximum 2-clique problem in unit-disk graphs that is based on geometric arguments is given in [53].

## 8 Concluding Remarks

Applications in large-scale social, telecommunication, biological, and transportation networks, where easy accessibility between the system's entities is of utmost importance, stimulated a significant activity in studying distance-based clique relaxation models,  $s$ -clique and  $s$ -club. This chapter presented an up-to-date survey of the literature of these models and the corresponding optimization problems. Due to its stronger cohesiveness properties and non-hereditary nature, which results in interesting research challenges, the maximum  $s$ -club problem has attracted much more attention. The lack of results for the maximum  $s$ -clique problem can also be explained by the fact that this problem is equivalent to the maximum clique problem in the corresponding power- $s$  graph, and the maximum clique problem has been studied extensively in the last several decades. While solving the maximum  $s$ -clique problem by reducing it to the maximum clique problem is, perhaps, most natural and straightforward approach, it is not clear whether it is most effective. The power- $s$  graph  $G^s$  typically has a much higher edge density than the original graph  $G$ , and the maximum clique problem is known to be particularly difficult to solve on dense graphs in practice. Moreover, clique is W[1]-hard, and, given that  $s$ -club is fixed-parameter tractable for  $s > 1$ , reducing  $s$ -clique to clique does not appear to be appealing from the parameterized complexity viewpoint. Therefore, investigating the maximum  $s$ -clique problem from alternative perspectives may be of interest.

**Acknowledgements** Partial support of Air Force Office of Scientific Research (Award FA9550-12-1-0103) and the U.S. Department of Energy (Award DE-SC0002051) is gratefully acknowledged. We thank Baski Balasundaram and Foad Mahdavi Pajouh for their comments on a preliminary version of the paper, which helped to improve the presentation.

## References

1. Abello, J., Resende, M.G.C., Sudarsky, S.: Massive quasi-clique detection. In: Rajsbaum, S. (ed.) LATIN 2002: Theoretical Informatics, pp. 598–612. Springer, London (2002)
2. Adler, N.: Competition in a deregulated air transportation market. *Eur. J. Oper. Res.* **129**, 337–345 (2001)
3. Alba, R.D.: A graph-theoretic definition of a sociometric clique. *J. Math. Sociol.* **3**, 113–126 (1973)
4. Almeida, M.T., Carvalho, F.D.: Integer models and upper bounds for the 3-club problem. *Networks* **60**, 155–166 (2012)
5. Arora, S., Lund, C., Motwani, R., Szegedy, M.: Proof verification and hardness of approximation problems. *J. ACM* **45**, 501–555 (1998)
6. Arora, S., Safra, S.: Probabilistic checking of proofs: a new characterization of NP. *J. ACM* **45**(1), 70–122 (1998)
7. Asahiro, Y., Miyano, E., Samizo, K.: Approximating maximum diameter-bounded subgraphs. In: Proceedings of the 9th Latin American Conference on Theoretical Informatics (LATIN'10), pp. 615–626. Springer, Berlin (2010)
8. Bader, J.S., Chaudhuri, A., Rothberg, J.M., Chant, J.: Gaining confidence in high-throughput protein interaction networks. *Nat. Biotechnol.* **22**, 78–85 (2004)

9. Balasundaram, B., Butenko, S.: Graph domination, coloring and cliques in telecommunications. In: Resende, M.G.C., Pardalos, P.M. (eds.) *Handbook of Optimization in Telecommunications*, pp. 865–890. Springer, New York (2006)
10. Balasundaram, B., Butenko, S., Trukhanov, S.: Novel approaches for analyzing biological networks. *J. Comb. Optim.* **10**, 23–39 (2005)
11. Batagelj, V.: Networks/Pajek graph files. <http://vlado.fmf.uni-lj.si/pub/networks/pajek/data/gphs.htm> (2005). Accessed July 2013
12. Berry, N., Ko, T., Moy, T., Smrcka, J., Turnley, J., Wu, B.: Emergent clique formation in terrorist recruitment. In: *The AAAI-04 Workshop on Agent Organizations: Theory and Practice*, July 25, 2004, San Jose, California (2004). <http://www.cs.uu.nl/~virginia/aotp/papers.htm>
13. Bollobás, B.: *Extremal Graph Theory*. Academic Press, New York (1978)
14. Bollobás, B.: *Random Graphs*. Academic Press, New York (1985)
15. Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M.: The maximum clique problem. In: Du, D.-Z., Pardalos, P.M. (eds.) *Handbook of Combinatorial Optimization*, pp. 1–74. Kluwer Academic, Dordrecht (1999)
16. Bourjolly, J.-M., Laporte, G., Pesant, G.: Heuristics for finding  $k$ -clubs in an undirected graph. *Comput. Oper. Res.* **27**, 559–569 (2000)
17. Bourjolly, J.-M., Laporte, G., Pesant, G.: An exact algorithm for the maximum  $k$ -club problem in an undirected graph. *Eur. J. Oper. Res.* **138**, 21–28 (2002)
18. Brélaz, D.: New methods to color the vertices of a graph. *Commun. ACM* **22**(4), 251–256 (1979)
19. Buchanan, A., Sung, J.S., Butenko, S., Boginski, V., Pasiliao, E.: On connected dominating sets of restricted diameter. Working paper (2012)
20. Butenko, S., Wilhelm, W.: Clique-detection models in computational biochemistry and genomics. *Eur. J. Oper. Res.* **173**, 1–17 (2006)
21. Carraghan, R., Pardalos, P.: An exact algorithm for the maximum clique problem. *Oper. Res. Lett.* **9**, 375–382 (1990)
22. Carvalho, F.D., Almeida, M.T.: Upper bounds and heuristics for the 2-club problem. *Eur. J. Oper. Res.* **210**(3), 489–494 (2011)
23. Chang, M.S., Hung, L.J., Lin, C.R., Su, P.C.: Finding large  $k$ -clubs in undirected graphs. In: *Proc. 28th Workshop on Combinatorial Mathematics and Computation Theory* (2011)
24. Chen, J., Huang, X., Kanj, I.A., Xia, G.: Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.* **72**, 1346–1367 (2006)
25. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. *Discrete Math.* **86**, 165–177 (1990)
26. Dekker, A., Pérez-Rosés, H., Pineda-Villavicencio, G., Watters, P.: The maximum degree & diameter-bounded subgraph and its applications. *J. Math. Model. Algorithms* **11**, 249–268 (2012)
27. Dekker, A.H.: Social network analysis in military headquarters using CAVALIER. In: *Proceedings of Fifth International Command and Control Research and Technology Symposium*, Canberra, Australia, pp. 24–26 (2000)
28. Diestel, R.: *Graph Theory*. Springer, Berlin (1997)
29. DIMACS. Second DIMACS Implementation Challenge. <http://dimacs.rutgers.edu/Challenges/> (1995). Accessed July 2013
30. DIMACS. Tenth DIMACS Implementation Challenge. <http://cc.gatech.edu/dimacs10/>. Accessed July 2013
31. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer, Berlin (1999)
32. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York (1979)
33. Gendreau, M., Soriano, P., Salvail, L.: Solving the maximum clique problem using a tabu search approach. *Ann. Oper. Res.* **41**, 385–403 (1993)
34. Goffman, C.: And what is your Erdős number? *Am. Math. Mon.* **76**, 791 (1969)
35. Grossman, J., Ion, P., De Castro, R.: The Erdős Number Project. <http://www.oakland.edu/enp/> (1995). Accessed July 2013

36. Hartung, S., Komusiewicz, C., Nichterlein, A.: Parameterized algorithmics and computational experiments for finding 2-clubs. In: Thilikos, D., Woeginger, G. (eds.) *Parameterized and Exact Computation. Lecture Notes in Computer Science*, vol. 7535, pp. 231–241. Springer, Berlin (2012)
37. Hartung, S., Komusiewicz, C., Nichterlein, A.: On structural parameterizations for the 2-club problem. In: *Proceedings of the 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM '13). Lecture Notes in Computer Science*, vol. 7741, pp. 233–243. Springer, Berlin (2013)
38. Jaillet, P., Song, G., Yu, G.: Airline network design and hub location problems. *Location Sci.* **4**, 195–212 (1996)
39. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) *Complexity of Computer Computations*, pp. 85–103. Plenum, New York (1972)
40. Kim, D., Wu, Y., Li, Y., Zou, F., Du, D.-Z.: Constructing minimum connected dominating sets with bounded diameters in wireless networks. *IEEE Trans. Parallel Distrib. Syst.* **20**(2), 147–157 (2009)
41. Luce, R.D.: Connectivity and generalized cliques in sociometric group structure. *Psychometrika* **15**, 169–190 (1950)
42. Luce, R.D., Perry, A.D.: A method of matrix analysis of group structure. *Psychometrika* **14**, 95–116 (1949)
43. Mahdavi Pajouh, F.: *Polyhedral combinatorics, complexity & algorithms for  $k$ -clubs in graphs*. PhD thesis, Oklahoma State University (July 2012)
44. Mahdavi Pajouh, F., Balasundaram, B.: On inclusionwise maximal and maximum cardinality  $k$ -clubs in graphs. *Discrete Optim.* **9**(2), 84–97 (2012)
45. Marinčec, J., Mohar, B.: On approximating the maximum diameter ratio of graphs. *Discrete Math.* **244**, 323–330 (2002)
46. Memon, N., Kristoffersen, K.C., Hicks, D.L., Larsen, H.L.: Detecting critical regions in covert networks: a case study of 9/11 terrorists network. In: *The Second International Conference on Availability, Reliability and Security*, pp. 861–870 (2007)
47. Miao, J., Berleant, D.: From paragraph networks to document networks. In: *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2004)*, vol. 1, pp. 295–302 (2004)
48. Mladenović, N., Hansen, P.: Variable neighborhood search. *Comput. Oper. Res.* **24**(11), 1097–1100 (1997)
49. Mokken, R.J.: Cliques, clubs and clans. *Qual. Quant.* **13**, 161–173 (1979)
50. Newman, M.: The structure and function of complex networks. *SIAM Rev.* **45**, 167–256 (2003)
51. Östergård, P.R.J.: A fast algorithm for the maximum clique problem. *Discrete Appl. Math.* **120**, 197–207 (2002)
52. Pasupuleti, S.: Detection of protein complexes in protein interaction networks using  $n$ -clubs. In: *Proceedings of the 6th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Lecture Notes in Computer Science*, vol. 4973, pp. 153–164. Springer, Berlin (2008)
53. Pattillo, J., Wang, Y., Butenko, S.: On distance-based clique relaxations in unit disk graphs. Working paper (2012)
54. Pattillo, J., Youssef, N., Butenko, S.: On clique relaxation models in network analysis. *Eur. J. Oper. Res.* (2012). doi:[10.1016/j.ejor.2012.10.021](https://doi.org/10.1016/j.ejor.2012.10.021)
55. Rothenberg, R.B., Potterat, J.J., Woodhouse, D.E.: Personal risk taking and the spread of disease: beyond core groups. *J. Infect. Dis.* **174**(Supp. 2), S144–S149 (1996)
56. Sageman, M.: *Understanding Terrorist Networks*. University of Pennsylvania Press, Philadelphia (2004)
57. Sampson, R.J., Groves, B.W.: Community structure and crime: testing social-disorganization theory. *Am. J. Sociol.* **94**, 774–802 (1989)
58. Schäfer, A.: *Exact algorithms for  $s$ -club finding and related problems*. Master's thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena (2009)

59. Schäfer, A., Komusiewicz, C., Moser, H., Niedermeier, R.: Parameterized computational complexity of finding small-diameter subgraphs. *Optim. Lett.* **6**, 883–891 (2012)
60. Scott, J.: *Social Network Analysis: A Handbook*, 2nd edn. Sage, London (2000)
61. Seidman, S.B.: Network structure and minimum degree. *Soc. Netw.* **5**, 269–287 (1983)
62. Seidman, S.B., Foster, B.L.: A graph theoretic generalization of the clique concept. *J. Math. Sociol.* **6**, 139–154 (1978)
63. Shahinpour, S., Butenko, S.: Algorithms for the maximum  $k$ -club problem in graphs. *J. Comb. Optim.* (2013). doi:[10.1007/s10878-012-9473-z](https://doi.org/10.1007/s10878-012-9473-z)
64. Terveen, L., Hill, W., Amento, B.: Constructing, organizing, and visualizing collections of topically related web resources. *ACM Trans. Comput.-Hum. Interact.* **6**, 67–94 (1999)
65. Veremyev, A., Boginski, V.: Identifying large robust network clusters via new compact formulations of maximum  $k$ -club problems. *Eur. J. Oper. Res.* **218**(2), 316–326 (2012)
66. Wasserman, S., Faust, K.: *Social Network Analysis*. Cambridge University Press, New York (1994)
67. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory Comput.* **3**, 103–128 (2007)

# GRASP with Path-Relinking for Facility Layout

R.M.A. Silva, M.G.C. Resende, P.M. Pardalos, G.R. Mateus, and G. De Tomi

**Abstract** This paper proposes a mathematical formulation for the facility layout problem (FLP) based on the generalized quadratic assignment problem (GQAP). The GQAP is a generalization of the NP-hard quadratic assignment problem (QAP) that allows multiple facilities to be assigned to a single location as long as the capacity of the location allows. As a case study, we adapt the GRASP with path-relinking (GRASP-PR) heuristic introduced in Mateus et al. (J. Heuristics 17:527–565, 2011) for the hospital layout problem (HLP). In the HLP, we are given a set of areas in a hospital where medical facilities, such as surgery and recovery rooms, can be located and a set of medical facilities, each facility with a required area, that must be located in the hospital. Furthermore, we are given a matrix specifying, for each ordered pair of facilities, the number of patients that transition from the first to the

---

R.M.A. Silva (✉)

Centro de Informática, Federal University of Pernambuco, Av. Jornalista Anibal Fernandes, s/n—Cidade Universitária, CEP 50740-560, Recife, PE, Brazil  
e-mail: [rmas@cin.ufpe.br](mailto:rmas@cin.ufpe.br)

M.G.C. Resende

Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932, USA  
e-mail: [mgcr@research.att.com](mailto:mgcr@research.att.com)

P.M. Pardalos

Department of Industrial and Systems Engineering, University of Florida, 303 Weil Hall, Gainesville, FL 32611, USA  
e-mail: [pardalos@ufl.edu](mailto:pardalos@ufl.edu)

G.R. Mateus

Dept. of Computer Science, Federal University of Minas Gerais, CEP 31270-010, Belo Horizonte, MG, Brazil  
e-mail: [mateus@dcc.ufmg.br](mailto:mateus@dcc.ufmg.br)

G. De Tomi

Departamento de Engenharia de Minas e de Petróleo, Escola Politécnica da Universidade de São Paulo, Av. Prof. Mello Moraes, 2373, Butantã, CEP 05508-030, São Paulo, SP, Brazil  
e-mail: [gdetomi@usp.br](mailto:gdetomi@usp.br)

second facility. The objective of the assignment is to minimize the total distance traveled by the patients. We illustrate our algorithm with a numerical example.

**Keywords** Hospital layout · Generalized quadratic assignment problem · GRASP · Path-relinking · Heuristics

## 1 Introduction

The facility layout problem (FLP) consists in assigning facilities to locations such that the total area of the facilities assigned to a location does not exceed the available area of the location. Among all feasible assignments, we seek one that minimizes the sum of products of flows between facility pairs and distances between locations to which the facility pairs are assigned. More formally, let  $N = \{1, \dots, n\}$  denote the set of facilities and  $M = \{1, \dots, m\}$  the set of locations. Furthermore, denote by  $A_{n \times n} = (a_{ii'})$  the flow between facilities  $i \in N$  and  $i' \in N$ , such that  $a_{ii'} \in \mathbb{R}^+$  if  $i \neq i'$  and otherwise  $a_{ii'} = 0$ , by  $B_{m \times m} = (b_{jj'})$  the distance between locations  $j \in M$  and  $j' \in M$ , such that  $b_{jj'} \in \mathbb{R}^+$  if  $j \neq j'$  and otherwise  $b_{jj'} = 0$ , and by  $C_{n \times m} = (c_{ij})$ , the cost of assigning facility  $i \in N$  to location  $j \in M$ , such that  $c_{ij} \in \mathbb{R}^+$ . Let  $z \in \mathbb{R}^+$  be a scaling factor called the unit traffic cost,  $q_i \in \mathbb{R}^+$  be the area demanded by facility  $i \in N$ , and  $Q_j \in \mathbb{R}^+$ , the total available area of location  $j \in M$ . The FLP can be modeled as a generalized quadratic assignment problem (GQAP). It consists in finding  $X_{n \times m} = (x_{ij})$ , with  $x_{ij} \in \{0, 1\}$ , where facility  $i \in N$  is assigned to location  $j \in M$  if and only if  $x_{ij} = 1$ , such that the constraints

$$\sum_{j \in M} x_{ij} = 1, \quad \forall i \in N, \quad (1)$$

$$\sum_{i \in N} q_i x_{ij} \leq Q_j, \quad \forall j \in M, \quad (2)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in N, \forall j \in M$$

are satisfied and the objective function

$$\sum_{i \in N} \sum_{j \in M} c_{ij} x_{ij} + z \sum_{i \in N} \sum_{j \in M} \sum_{i' \in N, i' \neq i} \sum_{j' \in M} a_{ii'} b_{jj'} x_{ij} x_{i'j'}$$

is minimized. Constraints (1) guarantee that each facility is assigned to exactly one location, while constraints (2) ensure that location capacities are not violated.

The facility layout problem (FLP) was first modeled as a quadratic assignment problem by Koopmans and Beckmann (1957) [7], as a linear integer programming problem by Lawler (1963) [10], as a quadratic set covering problem by Bazaraa (1975) [1], as a mixed integer programming problem by Kaufman and Broeckx (1978) [6], and as a graph theory problem by Foulds and Robinson (1976) [4] and Rosenblatt (1979) [15], among others (Kusiak and Heragu, 1987 [8]).

The main drawback of the quadratic assignment model is that it does not take into account that facilities can have different dimensions and that more than one facility can be assigned to a single location as long as the location can accommodate them. Approaches based on integer programming can only handle very small instances of the FLP. This paper models the FLP as a GQAP, thus satisfying this more realistic requirement.

We present a GRASP with path-relinking heuristic for the hospital layout problem (HLP), a FLP where medical facilities, such as intensive care units, surgery rooms, radiology rooms, and physician offices, must be feasibly located in a hospital. A numerical example shows the applicability of the proposed GRASP-PR heuristic for this hospital layout problem.

The paper is organized as follows. In Sect. 2, we review the GRASP with path-relinking procedure proposed in Mateus et al. (2011) [11]. The numerical example is described in Sect. 3. Concluding remarks are made in Sect. 4.

## 2 GRASP with Path-Relinking

A GRASP (Feo and Resende, 1989 [2]; 1995 [3]; Resende and Ribeiro, 2010 [13]) is a multi-start heuristic where at each iteration a greedy randomized solution is constructed to be used as a starting solution for local search. Local search repeatedly substitutes the current solution by a better solution in the neighborhood of the current solution. Each such replacement is called a *move*. If there is no better solution in the neighborhood, the current solution is declared a local minimum and the search stops. The best local minimum found over all GRASP iterations is output as the solution. One way to incorporate memory into GRASP is with path-relinking (Glover, 1996 [5]; Ribeiro and Resende, 2012 [14]). In GRASP with path-relinking (Laguna and Marti, 1999 [9]; Resende and Ribeiro, 2005 [12]), an elite set of diverse good-quality solutions is maintained to be used during each GRASP iteration. After a solution is produced with greedy randomized construction and local search, that solution is combined with a randomly selected solution from the elite set using the path-relinking operator. The combined solution is a candidate for inclusion in the elite set and is added to the elite set if it meets certain quality and diversity criteria.

Mateus et al. (2011) [11] introduced a GRASP with path-relinking for the GQAP. Algorithm 1 shows pseudo-code for this algorithm. The algorithm takes as input the set  $N$  of facilities, the set  $M$  of locations, the flow matrix  $A$ , the distance matrix  $B$ , the assignment cost matrix  $C$ , the scaling factor  $z$ , the facility demands  $q_i$ ,  $i \in N$ , and the location capacities  $Q_j$ ,  $j \in M$ , and outputs a feasible solution  $p^*$  specifying the location of each facility in the best solution found. After initializing the elite set  $P$  as empty, the GRASP with path-relinking iterations are computed until a stopping criterion is satisfied. This criterion could be, for example, a maximum number of iterations, a target solution quality, or a maximum number of iterations without improvement. During each iteration, a greedy randomized solution is generated and local search is applied using it as a starting point, resulting in a solution



**Data:**  $N, M, A, B, C, z, q_i, Q_j$ .  
**Result:** Feasible solution  $p^*$ .  
 $P \leftarrow \emptyset$ ;  
**while** *stopping criterion not satisfied* **do**  
     $p \leftarrow \text{GreedyRandomized}(\cdot)$ ;  
    **if** *elite set  $P$  has enough elements* **then**  
        **if**  $p$  *is not feasible* **then**  
            Randomly select a new solution  $p \in P$ ;  
        **end**  
         $p \leftarrow \text{LocalSearch}(p)$ ;  
        Randomly select a solution  $q \in P$   
         $r \leftarrow \text{PathRelinking}(p, q)$ ;  
        **if** *elite set  $P$  is full* **then**  
            **if**  $c(r) \leq \max\{c(s) \mid s \in P\}$  *and*  $r \not\approx P$  **then**  
                replace the element most similar to  $r$  among all  
                elements with cost worse than  $r$ ;  
            **end**  
            **else if**  $r \not\approx P$  **then**  
                 $P \leftarrow P \cup \{r\}$ ;  
            **end**  
        **else if**  $p$  *is feasible and*  $p \not\approx P$  **then**  
             $P \leftarrow P \cup \{p\}$ ;  
        **end**  
    **end**  
**return**  $p^* = \text{argmin}\{c(s) \mid s \in P\}$ ;

**Algorithm 1:** A GRASP with path-relinking heuristic for the FLP

$p$ . If the greedy randomized solution is infeasible, a feasible solution is randomly selected from the elite set and local search is applied on this solution. Path-relinking is applied between  $p$  and some elite solution  $q$  only if the elite set has at least a minimum number of elements. Otherwise, solution  $p$  is simply added to the elite set if it is sufficiently different from the solutions already in the elite set. To more precisely define the term *sufficiently different*, let the *symmetric difference*  $\Delta(x, y)$  between two solutions  $x$  and  $y$  be defined as the number of moves needed to transform  $x$  into  $y$  or vice-verse. For a given level of difference  $\delta$ , we say  $x$  is sufficiently different from all elite solutions in  $P$  if  $\Delta(x, p) > \delta$  for all  $p \in P$ , which we indicate by the notation  $x \not\approx P$ . If the elite set is not yet full, the solution  $r$  resulting from path-relinking is added to the elite set if  $r \not\approx P$ . Otherwise, if  $r$  is not of worse quality than any elite solution and  $r \not\approx P$ , then it will be added to the elite set in place of some elite solution. Among all elite solutions having cost no better than that of  $r$ , the one most similar to  $r$ , that is, with smallest symmetric difference with respect to  $r$ , is selected to be removed from the elite set. At the end, the best elite set solution is output as the solution of the GRASP with path-relinking heuristic.

## 2.1 Greedy Randomized Construction

The construction procedure builds a solution one assignment at a time. Suppose a partial solution is on hand, that is, a number of assignments have already been made. To make the next assignment, the procedure needs to select a new facility and a location. Locations are made available, one at a time. The procedure randomly determines whether to use a new location or a previously chosen location, favoring a new location when the previously chosen locations have insufficient or barely sufficient available capacity. If the procedure determines that a previously chosen location is to be selected, it then determines which facilities can be assigned to that location with the maximum available capacity and randomly selects one of these facilities to be assigned. Of the locations that can accommodate this facility, one is selected at random and the assignment is made. On the other hand, if there is no previously chosen location with sufficient capacity or if the available capacity is barely sufficient, a new location is selected at random from the set of yet unchosen locations.

The above procedure is not guaranteed to produce a feasible solution. The greedy randomized construction procedure, shown in Algorithm 2, addresses this difficulty by repeatedly applying the steps described above. The main loop in lines 1–21 is repeated a maximum number of times or until all facilities are assigned, that is, when  $F = \emptyset$ . In each iteration of the procedure, the working sets are initialized in line 2 and the threshold probability is set to 1 in line 3. The purpose of the threshold is to control whether a new location should be selected. Since it is initially set to 1, then in the first iteration of the until loop in lines 4–19, the procedure always selects a new (first) location. At each iteration of the until loop, the threshold is updated in line 17 such that it will be more likely that a new location is selected when there are few facilities that can be assigned to locations in the current set  $R$ . The until loop consists of two stages. With probability equal to the threshold, the first stage (lines 5–9) selects a new location in line 6, updates the sets  $L$  and  $CL$  in line 7, and in line 8 determines the set  $T$  of facilities that can be assigned to some selected location. In the second stage (lines 10–18), the procedure randomly selects a facility from set  $T$  in line 11, updates the sets  $T$ ,  $F$ , and  $CF$  in line 12, creates the location set  $R$  in line 13, randomly selects a location from that set in line 14, makes the assignment of the facility to the location in line 15, determines the set  $T$  of facilities that can be assigned to some selected location in line 16, and updates the threshold probability in line 17. The until loop is repeated until both sets  $T$  and  $L$  are empty in line 19. The while loop ends either with a valid assignment in line 25 (indicated by  $F = \emptyset$ ) or with no solution found determined in line 23.

## 2.2 Approximate Local Search

The construction procedure of Sect. 2.1 produces a feasible solution  $p$  that is not guaranteed to be locally optimal. A local search procedure is applied starting at  $p$

**Data:**  $\bar{t}$  = maximum number of tries  
**Result:** Solution  $x \in \mathcal{X}$

```

1 while  $k < \bar{t}$  and  $F \neq \emptyset$  do
2    $F \leftarrow N$ ;  $CF \leftarrow \emptyset$ ;  $L \leftarrow M$ ;  $CL \leftarrow \emptyset$ ;  $T \leftarrow \emptyset$ ;
3   Set threshold  $\leftarrow 1$ ;
4   repeat
5     if  $L \neq \emptyset$  and  $\text{random}([0, 1]) \leq \text{threshold}$  then
6       Randomly select a location  $l \in L$ ;
7       Update sets  $L \leftarrow L \setminus \{l\}$  and  $CL \leftarrow CL \cup \{l\}$ ;
8       Set  $T \subseteq F$  to be all facilities with demands less than or equal
       to the maximum slack in  $CL$ ;
9     end
10    if  $T \neq \emptyset$  then
11      Randomly select a facility  $f \in T$ ;
12      Update sets  $T \leftarrow T \setminus \{f\}$ ;  $F \leftarrow F \setminus \{f\}$ ; and
       $CF \leftarrow CF \cup \{f\}$ ;
13      Create set  $R \subseteq CL$  to be all locations having slack greater
      than or equal to demand of facility  $f$ ;
14      Randomly select a location  $l \in R$ ;
15      Assign facility  $f$  to location  $l$ ;
16      Set  $T \subseteq F$  to be all facilities with demands less than or equal
      to the maximum slack in  $CL$ ;
17      Set threshold  $\leftarrow 1 - |T|/|F|$ ;
18    end
19    until  $T = \emptyset$  and  $L = \emptyset$ ;
20     $k \leftarrow k + 1$ ;
21  end
22  if  $F \neq \emptyset$  then
23    Solution not found;
24  else
25    return assignment  $x \in \mathcal{X}$ ;
26  end

```

**Algorithm 2:** Pseudo-code for GreedyRandomized: Greedy randomized construction procedure

to find an approximate local minimum. The local search procedure makes use of two neighborhood structures which we call *1-move* and *2-move*. A solution in the 1-move neighborhood of  $p$  is obtained by changing one facility-to-location assignment in  $p$ . Likewise, a solution in the 2-move neighborhood of  $p$  is obtained by simultaneously changing two facility-to-location assignments in  $p$ .

One way to carry out a local search in these neighborhoods is to evaluate moves in the 1-move neighborhood and move to the first improving solution. If no 1-move improving solution exists, 2-move neighborhood solutions are evaluated and a move is made to the first improving solution. Another way to carry out the local search is to evaluate all 1-move and 2-move neighborhood solutions

**Data:**  $\pi, MaxCLS, MaxItr$   
**Result:** Approximate local minimum  $\pi$

```

1 repeat
2    $count \leftarrow 0; CLS \leftarrow \emptyset;$ 
3   repeat
4      $\pi' \leftarrow \text{Move}(\pi);$ 
5     if  $\pi'$  is feasible and  $cost(\pi') < cost(\pi)$  then
6        $CLS \leftarrow CLS \cup \{\pi'\};$ 
7     end
8      $count \leftarrow count + 1;$ 
9   until  $|CLS| \geq MaxCLS$  or  $count \geq MaxItr;$ 
10  if  $CLS \neq \emptyset$  then
11    Randomly select a solution  $\pi \in CLS;$ 
12  end
13 until  $CLS = \emptyset;$ 
14 return  $\pi;$ 

```

**Algorithm 3:** Pseudo-code for ApproxLocalSearch: Approximate local search procedure

and move to the best improving solution. In both variants, the search is repeated until no improving solution in the neighborhoods exists. We propose a trade-off approach here. Instead of evaluating all of the 1-move and 2-move neighborhood solutions, we sample these neighborhoods and populate a candidate list with improving solutions. One of the solutions from the candidate list is randomly selected and a move is made to that solution. The search is repeated until no improving solution is sampled. Because solutions are sampled, not all neighbors may be evaluated. Consequently, the best solution found may not be a local minimum. Mateus et al. (2011) [11] call this solution an *approximate local minimum*.

Pseudo-code for the approximate local search is shown in Algorithm 3. The procedure takes as input the starting solution  $\pi$  and two parameters,  $MaxCLS$  and  $MaxItr$ , which control the sampling. The repeat until loop in lines 1–13 is repeated until an approximate local minimum is produced. In line 2, the sampling counter  $count$  and the candidate list  $CLS$  are initialized. At each iteration of the inner loop in lines 3–9, the 1-move and 2-move neighborhoods of  $\pi$  are sampled without replacement by procedure  $\text{Move}(\pi)$  in line 4. If this neighbor is an improving feasible solution, it is inserted into  $CLS$  in line 6. This is repeated until either the candidate list is full or a maximum number of neighbors have been sampled. In lines 10–12, if the candidate list is not empty, an assignment  $\pi \in CLS$  is randomly chosen. If the set  $CLS$  is empty after the sampling process, the procedure terminates returning  $\pi$  as an approximate local minimum in line 14. Otherwise, the procedure moves to a solution in  $CLS$ , repeating the outer loop.

### 2.3 Path-Relinking

Motivated by the fact that a single move from a solution  $x$  in the direction of a target solution  $x_t$  does not guarantee the feasibility of the new constructed solution, a new variant of path-relinking is proposed in Mateus et al. (2011) [11].

Suppose that among the differences between  $x$  and  $x_t$  is the location assigned to facility  $f$ . In other words, while the location assigned to  $f$  in  $x_t$  is  $l$ , the location assigned to  $f$  in  $x$  is  $l'$ , with  $l \neq l'$ . In this case, is not necessarily feasible to perform a move in  $x$  that assigns  $f$  to  $l$ . If the capacity  $Q_l$  is not violated, then the new solution is feasible. Otherwise, a repair procedure must be applied to try to make it feasible.

In this repair procedure, a facility set  $F$  is created with all not yet fixed facilities assigned to location  $l$  for which capacity is violated. Next, the set  $T \subseteq F$  is constructed with all facilities in  $F$  having demands less than or equal to the maximum available capacity of locations in  $M$ . After a facility from  $T$  is randomly selected, set  $R$  consists of locations in  $M$  that can accommodate it. A location is selected from set  $R$  and the facility is assigned to it. This process is repeated until the capacity of location  $l$  has a nonnegative slack.

The path-relinking process is a sequence of steps from  $x_s$  to  $x_t$ . In each step, a move is performed from the current solution  $x$  with or without repair. Next, a facility  $i$  is randomly selected from a set composed of all not yet fixed facilities corrected in the step. A facility is *corrected* when its location becomes the same as the one assigned to it in the target solution  $x_t$ . After facility  $i$  is fixed, the next step begins. This process continues until the target solution  $x_t$  is reached or when no feasible solution is obtained from  $x$ .

Algorithm 4 shows pseudo-code for the path-relinking procedure. The algorithm takes as input  $\pi_s$  and  $\pi_t$ , the starting and target solutions, respectively, and outputs the best solution  $\pi^*$  in path from  $\pi_s$  to  $\pi_t$ . Initially, the best solution in the path is set as  $\pi^*$  in line 1 and its corresponding objective function is assigned to  $f^*$  in line 2. In line 3, the current solution  $\pi'$  is initialized with  $\pi_s$ , and the working sets *Fix* and *nonFix* are respectively initialized empty and with  $N$ . The while loop in lines 5–35 is repeated until all facilities in  $\pi'$  are assigned to the same locations assigned to them in  $\pi_t$ , that is, the set  $\varphi(\pi', \pi_t) = \{i \in N \mid \pi'(i) \neq \pi_t(i)\}$  is empty, where  $\pi'(i)$  and  $\pi_t(i)$  are the locations assigned to facility  $i$  in solutions  $\pi'$  and  $\pi_t$ , respectively.

After the set  $\mathcal{B}$  of best solutions is set to empty in line 6, each while loop iteration consists of two stages. The first stage in lines 7–20 implements the path-relinking step. It creates set  $\mathcal{B}$  with the best feasible solutions constructed from the current solution  $\pi'$ . In line 6,  $\mathcal{B}$  is initialized as empty. Each facility  $v \in \varphi(\pi', \pi_t)$  is analyzed in lines 7–20 to create the set  $\mathcal{B}$  with the best feasible solutions constructed from the current solution  $\pi'$ . Procedure `makeFeasible` is applied in line 8 to facility  $v$  to attempt to create a new solution  $\bar{\pi}$  from  $\pi'$ . The application of `makeFeasible` to facility  $v$  can either result in a feasible or infeasible solution. In case `makeFeasible` returns a feasible solution  $\bar{\pi} \notin \mathcal{B}$ ,  $\bar{\pi}$  is added to  $\mathcal{B}$  if  $\mathcal{B}$  is not yet full. Otherwise, if  $\mathcal{B}$  is full and solution  $\bar{\pi} \notin \mathcal{B}$  is not worse than any elite solution, then  $\bar{\pi}$  is added to  $\mathcal{B}$  replacing some other elite solution.

**Data:** Starting solution  $\pi_s$ , target solution  $\pi_t$ , and candidate size factor  $\eta$   
**Result:** Best solution  $\pi^*$  in path from  $\pi_s$  to  $\pi_t$






```

1  $\pi^* \leftarrow \operatorname{argmin}\{f(\pi_s), f(\pi_t)\};$ 
2  $f^* \leftarrow f(\pi^*);$ 
3  $\pi' \leftarrow \pi_s; \text{Fix} \leftarrow \emptyset; \text{nonFix} \leftarrow N;$ 
4 Compute difference  $\varphi(\pi', \pi_t)$  between solution  $\pi'$  and  $\pi_t$ ;
5 while  $\varphi(\pi', \pi_t) \neq \emptyset$  do
6    $\mathcal{B} \leftarrow \emptyset;$ 
7   for  $\forall v \in \varphi(\pi', \pi_t)$  do
8     Move the facility  $v$  in  $\pi'$  to the same location  $l$  assigned to  $v$  in
       $\pi_t$ ;
9      $\tilde{\pi} \leftarrow \text{makeFeasible}(\pi', v);$ 
10    if  $\tilde{\pi}$  is feasible then
11      if  $|\mathcal{B}| \geq \eta \cdot |\varphi(\pi', \pi_t)|$  then
12        if  $c(\tilde{\pi}) \leq \max\{c(\pi) \mid \pi \in \mathcal{B}\}$  and  $\tilde{\pi} \notin \mathcal{B}$  then
13          replace the element most similar to  $\tilde{\pi}$  among all
            elements with cost worst than  $\tilde{\pi}$ ;
14        end
15      else if  $\tilde{\pi} \notin \mathcal{B}$  then
16         $\mathcal{B} \leftarrow \mathcal{B} \cup \{\tilde{\pi}\};$ 
17      end
18    end
19  end
20  if  $\mathcal{B} \neq \emptyset$  then
21    Randomly select a solution  $\pi \in \mathcal{B}$ ;
22    Compute difference  $\varphi(\pi, \pi_t)$  between solution  $\pi$  and  $\pi_t$ ;
23    Set  $I = \varphi(\pi', \pi_t) \setminus (\varphi(\pi', \pi_t) \cap \varphi(\pi, \pi_t))$ ;
24    Randomly select a facility  $i \in I$ ;
25     $\text{Fix} \leftarrow \text{Fix} \cup \{i\}; \text{nonFix} \leftarrow \text{nonFix} \setminus \{i\};$ 
26     $\pi' \leftarrow \pi;$ 
27    if  $f(\pi') < f^*$  then
28       $f^* \leftarrow f(\pi');$ 
29       $\pi^* \leftarrow \pi';$ 
30    end
31  else
32    return assignment  $\pi^*$ ;
33  end
34 end
35 return assignment  $\pi^*$ ;
```

**Algorithm 4:** Pseudo-code for PathRelinking: Path-relinking procedure

In the second stage (lines 21–34), the procedure first randomly selects a solution  $\pi$  from set  $\mathcal{B}$  in line 22. Then, in line 25, it selects at random a facility  $i \in I = \varphi(\pi', \pi_t) \setminus (\varphi(\pi', \pi_t) \cap \varphi(\pi, \pi_t))$ , where  $I$  is defined in line 24 as the set containing all unfixed facilities whose locations were corrected in the previous path-relinking

**Fig. 1** Medical facilities: types, quantities, and dimensions

	ICU ::: quantity: 3 ::: area 135 m <sup>2</sup>
	Pediatric ICU ::: quantity: 6 ::: area 110 m <sup>2</sup>
	Operating room ::: quantity: 9 ::: area 90 m <sup>2</sup>
	Radiology ::: quantity: 12 ::: area 65 m <sup>2</sup>
	Physician's office ::: quantity: 15 ::: area 45 m <sup>2</sup>

step. A facility is corrected when its location becomes the one assigned to it in the target solution. After fixing facility  $i \in I$ , sets *Fix* and *nonFix* are updated in line 26. Finally, the next path-relinking step solution  $\pi'$  is set as  $\pi$  in line 27 and, if  $f(\pi') < f^*$ , then the best cost  $f^*$  and best solution  $\pi^*$  are updated in lines 29–30, respectively. However, if in some path-relinking step no feasible solution is obtained from  $\pi'$ , the while loop is interrupted, returning the current solution  $\pi^*$  as the result in line 33. If the target solution is reached, then  $\pi^*$  is returned in line 36.

This path-relinking is different from the standard variant because given solutions  $x_s$  and  $x_t$ , their common elements are not kept fixed a priori, such that a small portion of the solution space spanned by the remaining elements is explored. The new variant fixes one facility at time at each step.

### 3 An Illustrative Example of Hospital Layout

In this section, we provide an example to illustrate a hospital layout problem and show a corresponding solution produced with the GRASP with path-relinking heuristic described in Sect. 2.

#### 3.1 The Layout Problem Instance

Our example considers the medical departments listed in Fig. 1: three intensive-care units (ICUs), six pediatric intensive-care units (PICUs), nine operating (OR) rooms, twelve radiology (R) rooms, and fifteen physician offices (PO). Table 1 shows the people flow for each pair of facilities.

Figure 2 presents a hypothetical plant for a hospital with three floors and an elevator. The first floor plant has four locations, numbered 1–4, with dimensions 400 m<sup>2</sup>, 1050 m<sup>2</sup>, 450 m<sup>2</sup>, and 600 m<sup>2</sup>, respectively. The second floor has four locations, numbered 5–8, with 250 m<sup>2</sup>, 1000 m<sup>2</sup>, 375 m<sup>2</sup>, and 875 m<sup>2</sup>, respectively. The third floor has four locations, numbered 9–12, with 400 m<sup>2</sup>, 1050 m<sup>2</sup>, 450 m<sup>2</sup>, and 600 m<sup>2</sup>, respectively.

**Table 1** Unsymmetrical people flow for each pair of facilities

	ICU	PICU	OR	R	PO
Intensive care unit (ICU)	0	70	500	90	100
Pediatric ICU (PICU)	90	0	300	78	95
Operating room (OR)	700	250	0	200	50
Radiology (R)	80	60	300	0	380
Physician’s office (PO)	200	150	600	800	0

**Fig. 2** Hospital plant with three floors and twelve locations

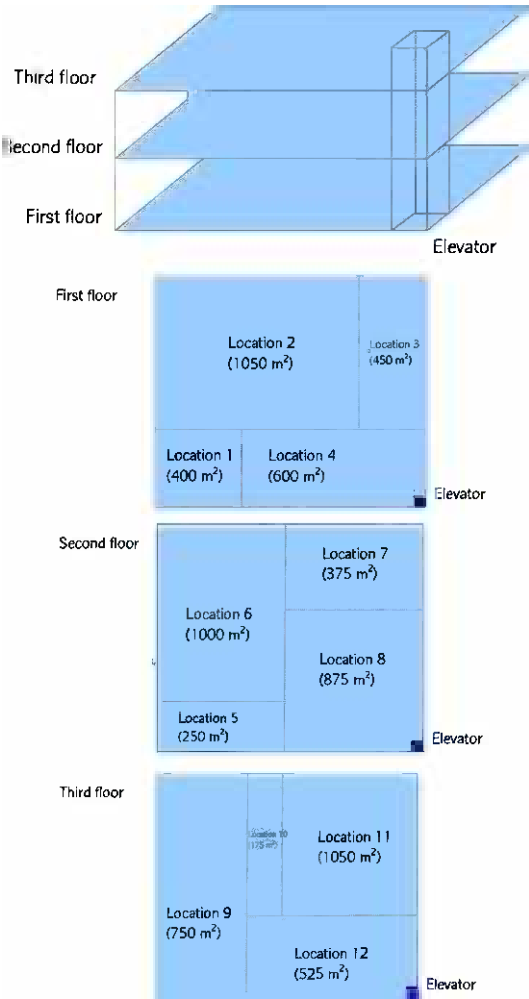


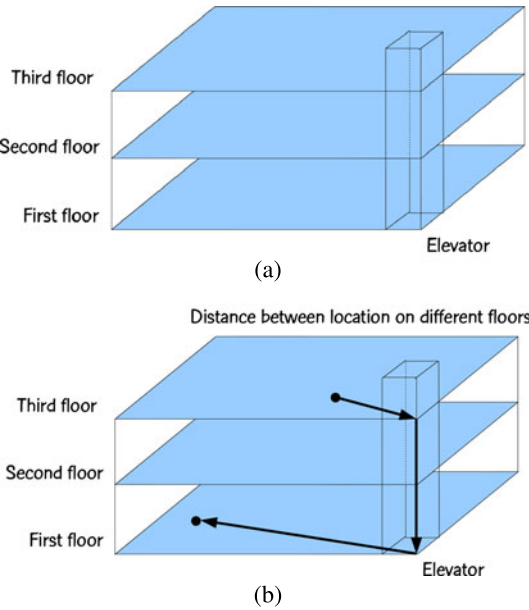
Table 2 shows the Euclidean distances between each of the 12 hospital locations. Distances between pairs of locations on same floor are assumed to be the Euclidean distances between the centers of the locations (Fig. 3(a)). Distances between lo-



**Table 2** Symmetrical distances between hospital locations

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	26.1	41	25	99.06	109.23	105.53	82.73	130.53	127.19	117.02	100.23
2	26.1	0	25	30.25	105.99	115.76	112.06	89.26	137.06	133.72	123.55	106.76
3	41	25	0	26.1	93.62	103.79	100.09	77.29	125.09	121.75	111.58	94.79
4	25	30.25	26.1	0	75.83	86	82.3	59.5	107.3	103.96	93.79	77
5	99.06	105.99	93.62	75.83	0	25	45.07	27.95	107.13	103.79	93.62	76.83
6	109.23	115.76	103.79	86	25	0	27.95	27.95	117.3	115.26	103.79	87
7	105.53	112.06	100.09	82.3	45.07	27.95	0	25	113.6	110.26	100.09	83.3
8	82.73	89.26	77.29	59.5	27.95	27.95	25	0	90.8	87.46	77.29	60.5
9	130.53	137.06	125.09	107.3	107.13	117.3	113.6	90.8	0	12.5	28.5	30.51
10	127.19	133.72	121.75	103.96	103.79	115.26	110.26	87.46	12.5	0	17.5	29.1
11	117.02	123.55	111.58	93.79	93.62	103.79	100.09	77.29	28.5	17.5	0	25.12
12	100.23	106.76	94.79	77	76.83	87	83.3	60.5	30.51	29.1	25.12	0

**Fig. 3** Distance between locations



**Table 3** Facility-location assignment cost

	1	2	3	4	5	6	7	8	9	10	11	12
ICU	1200	1050	1200	1300	1200	1050	1200	1300	1200	1050	1200	1300
PICU	1000	850	1000	1100	1000	850	1000	1100	1000	850	1000	1100
OR	800	650	800	900	800	650	800	900	800	650	800	900
R	600	450	600	700	600	450	600	700	600	450	600	700
PO	400	250	400	500	400	250	400	500	400	250	400	500

cations on different floors are assumed to be the sums of the Euclidean distance between the center of the first location to the elevator on that floor, the distance traveled by elevator, and the Euclidean distance between the elevator on the other floor and the center of the second location (Fig. 3(b)).

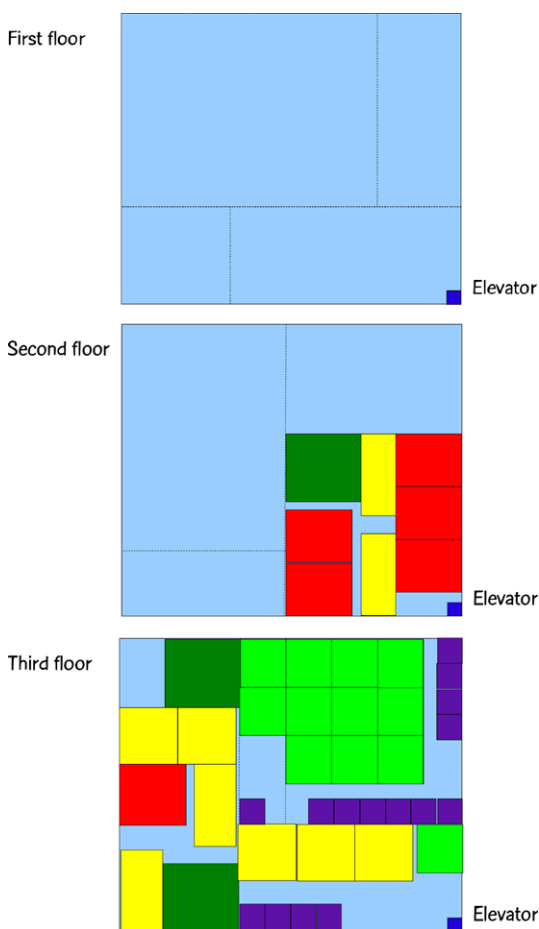
Finally, Table 3 shows the cost of assigning each medical facility to each of the twelve locations illustrated in Fig. 2.

**3.2 The Solution Found by GRASP with Path-Relinking**

Figure 4 shows the assignment found with GRASP with path-relinking heuristic. We make the following observations regarding the solution.

- Since all the facilities could be accommodated in two floors, only two floors were used.

**Fig. 4** Assignment found by GRASP-PR in 1898.4 seconds on a 2.6 GHz machine



- The heuristic assigned facilities to floors 2 and 3. However, because of the symmetry of floors 1 and 3, a solution with equal cost would allocate all third floor facilities to the first floor.
- On the second floor, the facilities are concentrated around the elevator since patients often move between second floor ICUs and PICUs and third floor radiology facilities.
- The ICUs were located near the ORs since their inter-facility flows were high.
- Note that a single PICU was located on the third floor while a single ICU was located on the second. All radiology facilities are located on the third floor. Even though the flow between the ICU and radiology facilities is greater than the flow between PICU and radiology facilities, the lone ICU on the second floor had to be placed there since there was insufficient space for it in any location on the third floor. The lone PICU on the third floor was located there and not on the second floor since it is near the radiology facilities.

## 4 Concluding Remarks

In this paper, we revisited the GRASP with path-relinking heuristic for the generalized quadratic assignment problem (GQAP) of Mateus et al. (2011) [11] and applied the heuristic to solve a facility layout problem (FLP) as a generalized quadratic assignment problem (GQAP). We illustrate the solution method with a hypothetical hospital layout problem. To implement this approach, we require inter-facility flow rates. These will need to be established from data gathered in other hospital settings and adapted to the setting under consideration in the design. The approach can also be applied to other settings, such as sports facilities, shopping malls, and airports.

**Acknowledgements** The research of R.M.A Silva was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq), the Foundation for Support of Research of the State of Minas Gerais, Brazil (FAPEMIG), Coordination for the Improvement of Higher Education Personnel, Brazil (CAPES), Foundation for the Support of Development of the Federal University of Pernambuco, Brazil (FADE), the Office for Research and Graduate Studies of the Federal University of Pernambuco (PROPESQ), and the Foundation for Support of Science and Technology of the State of Pernambuco (FACEPE).

## References

1. Bazaraa, M.S.: Computerized layout design: a branch and bound approach. *AIIE Trans.* **7**(4), 432–438 (1975)
2. Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. *Oper. Res. Lett.* **8**, 67–71 (1989)
3. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. *J. Glob. Optim.* **6**, 109–133 (1995)
4. Foulds, L.R., Robinson, D.F.: A strategy for solving the plant layout problem. *Oper. Res. Q.* **27**(4), 845–855 (1976)
5. Glover, F.: Tabu search and adaptive memory programming—advances, applications and challenges. In: Barr, R.S., Helgason, R.V., Kennington, J.L. (eds.) *Interfaces in Computer Science and Operations Research*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 1–75. Kluwer Academic, Dordrecht (1996)
6. Kaufman, L., Broeckx, F.: An algorithm for the quadratic assignment problem using Bender's decomposition. *Eur. J. Oper. Res.* **2**(3), 207–211 (1978)
7. Koopmans, T.C., Beckmann, M.: Assignment problems and the location of economic activities. *Econometrica* **25**(1), 53–76 (1957)
8. Kusiak, A., Heragu, S.S.: The facility layout problem. *Eur. J. Oper. Res.* **29**(3), 229–251 (1987)
9. Laguna, M., Marti, R.: GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. Comput.* **11**, 44–52 (1999)
10. Lawler, E.L.: The quadratic assignment problem. *Manag. Sci.* **9**(4), 586–599 (1963)
11. Mateus, G.R., Resende, M.G.C., Silva, R.M.A.: GRASP with path-relinking for the generalized quadratic assignment problem. *J. Heuristics* **17**, 527–565 (2011)
12. Resende, M.G.C., Ribeiro, C.C.: GRASP with path-relinking: recent advances and applications. In: Ibaraki, T., Nonobe, K., Yagiura, M. (eds.) *Metaheuristics: Progress as Real Problem Solvers*, pp. 29–63. Springer, Berlin (2005)

13. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures: advances and applications. In: Gendreau, M., Potvin, J.Y. (eds.) *Handbook of Metaheuristics*, 2nd edn., pp. 293–319. Springer, Berlin (2010)
14. Ribeiro, C.C., Resende, M.G.C.: Path-relinking intensification methods for stochastic local search algorithms. *J. Heuristics* **18**, 193–214 (2012)
15. Rosenblatt, M.J.: The facilities layout problem: a multigoal approach. *J. Prod. Res.* **17** (1979)

# Comparative Analysis of the BRIC Countries Stock Markets Using Network Approach

Arsenii Vizgunov, Andrey Glotov, and Panos M. Pardalos

**Abstract** The paper presents the analysis of the network model referred to as market graph of the BRIC countries stock markets. We construct the stock market graph as follows: each vertex represents a stock, and the vertices are adjacent if the price correlation coefficient between them over a certain period of time is greater than or equal to specified threshold. The market graphs are constructed for different time periods to understand the dynamics of their characteristics such as correlation distribution histogram, mean value and standard deviation, size and structure of the maximum cliques. Our results show that we can split the BRIC countries into two groups. Brazil, Russia and India constitute the first group, China constitutes the second group.

**Keywords** Comparative analysis · BRIC · Stock market · Network approach · Market graph

## 1 Introduction

The modern IT technologies provide the possibility to invest in different countries stock markets. In order to manage the complexity of the datasets generated by the stock markets investors need the mathematical models to present the stock market as a set of their structural properties. The analysis of the dynamics of these properties provides investors with the important information about similarity and peculiarity of the prospective investment stock markets. In our paper we use the network model

---

A. Vizgunov (✉) · A. Glotov

National Research University Higher School of Economics, 25/12 Bolshaja Pecherskaja Ulitsa,  
Nizhny Novgorod, 603155, Russian Federation  
e-mail: [anvizgunov@hse.ru](mailto:anvizgunov@hse.ru)

A. Glotov

e-mail: [aaglotov@edu.hse.ru](mailto:aaglotov@edu.hse.ru)

P.M. Pardalos

Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA  
e-mail: [pardalos@ise.ufl.edu](mailto:pardalos@ise.ufl.edu)

introduced by Boginski et al. (2003) [1] referred to as market graph. Each vertex of the market graph model represents a stock. The correlation coefficient of the stock returns is used as a measure of the similarity of the stocks. The edge exists between the vertices if the corresponding correlation coefficient is equal to or greater than the specified threshold.

The market graph model is widely used. In particular it was used to analyze US (Boginski et al. 2003 [1], 2005 [2], 2006 [3]), Chinese (Huang et al. 2009) [4], Swedish (Jallo et al. 2012) [5], Russian stock markets (Vizgunov et al. 2012) [8]. The most of papers describe a particular country stock market and sometimes compare calculated characteristics with the characteristics of the well developed US stock market. We extend this approach and use market graph model as a basis to compare several BRIC countries markets. The importance of the BRIC countries for the world economy increases over time that justifies the hypothesis about the similarity of the structural properties dynamics of the stock markets. In order to have accurate comparison results, we consider different countries market graphs with the same density and use similar quantity of stocks of the biggest enterprises in terms of capitalization for each country. We use the following characteristics for our study: trades intensity, correlation coefficients distribution histogram, mean value and standard deviation, dependency of the market graph density on the threshold value, size and structure of the maximum cliques.

All considered stock markets are similar from the trades' intensity point of view but if we take into account other characteristics the stock markets can be split into two groups. The Brazilian, Russian and Indian stock markets are similar to each other but the Chinese stock market differs.

The paper is organized as follows. In Sect. 2, we briefly recall the market graph model and describe the Brazilian, Russian, Indian and Chinese stock market data. In Sect. 3 and Sect. 4, we present the results of the analysis of the network models and market graph models accordingly. In Sect. 4, we give concluding remarks.

## 2 Market Graph Construction

### 2.1 Network and Market Graph Models

The basis of the model is the market network constructed using correlation as a measure of similarity between stocks (Mantegna and Stanley 2000 [6], Salter-Townshend M. 2012 [7]). Each vertex represents a stock and is connected with all others vertices. Correlation coefficient value is associated with corresponding edge. More precisely, let  $P_i(t)$  be the price of stock  $i$  on day  $t$  ( $i = 1, \dots, N; t = 1, \dots, n$ ). Then

$$R_i(t) = \ln \left( \frac{P_i(t)}{P_i(t-1)} \right) \quad (1)$$

defines the return of the stock  $i$  over the one-day period from  $(t - 1)$  to  $t$ . The average return of the stock  $i$  over the period of  $n$  days is defined by

$$E(R_i) = \frac{1}{n} \sum_{t=1}^n R_i(t) \quad (2)$$

The variance of the stock  $i$  over the period of  $n$  days is defined by

$$\text{var}(R_i) = \frac{1}{n} \sum_{t=1}^n (R_i(t) - E(R_i))^2 \quad (3)$$

The correlation coefficient  $c_{ij}$  of two stocks  $i$  and  $j$  ( $i = 1, \dots, N$ ;  $j = 1, \dots, N$ ) over  $n$  days is defined by

$$c_{ij} = \frac{E(R_i R_j) - E(R_i)E(R_j)}{\sqrt{\text{var}(R_i) \text{var}(R_j)}} \quad (4)$$

To construct the market graph model, we should choose the threshold value. Two vertices  $i$  and  $j$  are connected by an edge in the market graph if the corresponding correlation coefficient is larger than or equal to the specified threshold  $\theta \in [-1, 1]$ .

A clique in a graph is a subset of its vertices such that every two vertices in the subset are connected by an edge. A maximum clique is a clique of the largest possible size in a given graph. In the market graph a clique is formed by jointly pair wise correlated group of stocks.

## 2.2 Data of the Brazilian, Russian, Indian and Chinese Stock Markets

In order to construct the market graphs, we consider the 200 stocks of the biggest in terms of capitalization enterprises of Brazil, India and China and 270 stocks of the biggest Russian enterprises traded on MICEX SE from 2007 to 2011. The data was taken from the web site of the Emerging Market Information Service EMIS (<http://www.securities.com/products/emis.html>) and web site of the Moscow Interbank Currency Exchange Stock Exchange (<http://www.micex.ru/marketdata/archive>).

To analyze the dynamics of the characteristics change over the time, we consider fifteen shifts within considered period. Each shift has 500 trading days of the Moscow Interbank Currency Exchange Stock Exchange. The starting points of every two consecutive shifts are separated by the interval of 50 days. Therefore, every pair of consecutive shifts has 450 days in common and 50 days different. For Brazil, India and China, the starting and ending dates are the same as they are for Russia. The quantity of the trading days of one shift for Brazil, India and China varies from 483 to 502. Dates corresponding to each shift and quantity of the trading days within each period for each country are summarized in Table 1. Note that the periods 1–10 include the financial crisis of 2008.



**Table 1** Dates corresponding to considered 500-day shifts

#	Starting date	Ending date	Brazil	Russia	India	China
1	01/09/2007	01/16/2009	500	500	501	492
2	03/22/2007	03/31/2009	501	500	500	492
3	06/04/2007	06/11/2009	499	500	498	493
4	08/14/2007	08/21/2009	500	500	497	493
5	10/23/2007	10/30/2009	500	500	495	492
6	01/11/2008	01/19/2010	500	500	491	492
7	03/25/2008	04/01/2010	501	500	491	492
8	06/06/2008	06/15/2010	493	500	493	491
9	08/18/2008	08/24/2010	492	500	493	490
10	10/29/2008	11/02/2010	486	500	493	487
11	01/19/2009	01/21/2011	487	500	497	489
12	04/01/2009	04/05/2011	483	500	499	487
13	06/15/2009	06/17/2011	485	500	502	489
14	08/24/2009	08/26/2011	485	500	501	489
15	11/02/2009	11/07/2011	485	500	501	490

### 3 Analysis of the Network Models

One of the common characteristic of the considered emerging stock markets is the existence of the trading days without transactions for almost all stocks. It demands using a cleaning procedure to smooth data. We smooth the data in the following way: when there were no transactions on a given trading day the price for a stock was considered to be unchanged. If a stock was offered for trading after the beginning of the studied period, then the price was considered to be equal to the price fixed on the first day of trading. When there were no transactions on a given trading day, the volume for a stock was considered to be equal to zero.

Before applying a smoothing procedure we decrease the need for adjusting data by means of excluding the stocks with large number of days without transactions. Table 2 shows the dependency of the considered stocks number on the amount of days when a stock should be transacted.

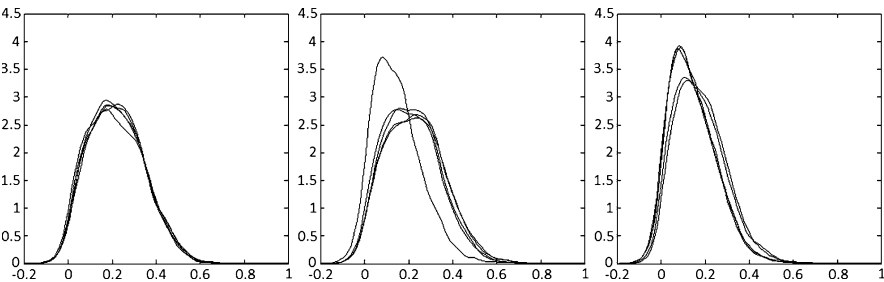
We try to take into consideration as many stocks as possible but at the same time we want to decrease needs to adjust data regarding to the days without trading transactions. Ultimately we use such stocks where the number of the days when a stock has been transacted exceeds 300 days out of the 500 trading days under consideration.

We would like to note that trades' intensity increases over the time for all countries stock markets.

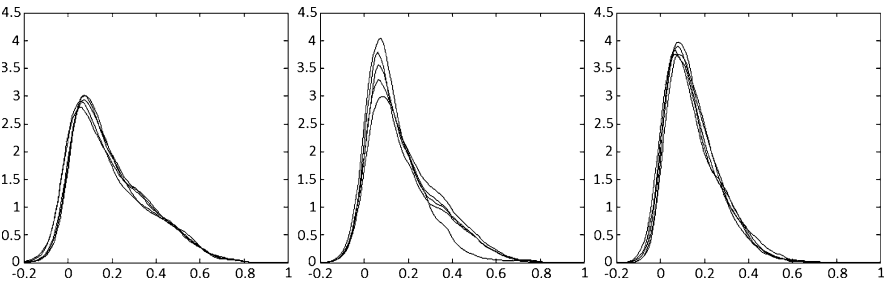
Figures 1, 2, 3 and 4 show the correlation distribution histogram for Brazilian, Russian, Indian and Chinese stock markets. Table 3 shows mean correlation and standard deviations values for different periods. The analysis of Figs. 1–4 and Table 3 shows that Brazilian and Indian correlation histogram curves have similar

**Table 2** Dependency of the considered stocks number on the amount of days when a stock should be transacted

#	Brazil			Russia			India			China		
	200	300	400	200	300	400	200	300	400	200	300	400
1	135	116	84	171	143	114	182	178	173	149	128	52
2	140	122	97	178	154	122	184	180	176	153	132	55
3	144	125	104	184	154	120	184	182	178	155	140	62
4	143	130	114	194	160	125	184	184	180	160	150	68
5	143	131	117	209	164	129	184	184	181	161	152	90
6	143	133	120	217	171	135	184	184	184	161	156	123
7	145	134	125	224	184	143	185	184	184	161	159	138
8	145	134	124	227	192	151	185	184	184	161	159	143
9	148	135	125	225	198	165	189	185	184	165	160	153
10	150	136	124	231	202	174	191	185	184	169	160	155
11	152	140	124	233	208	181	192	189	185	177	165	156
12	156	144	124	239	216	188	193	191	186	180	166	157
13	158	147	128	242	221	190	193	192	189	182	174	159
14	161	153	130	240	228	185	195	193	191	186	179	163
15	164	154	133	238	231	183	195	193	192	187	180	171



**Fig. 1** The correlation distribution for Brazilian stock market



**Fig. 2** The correlation distribution for Russian stock market

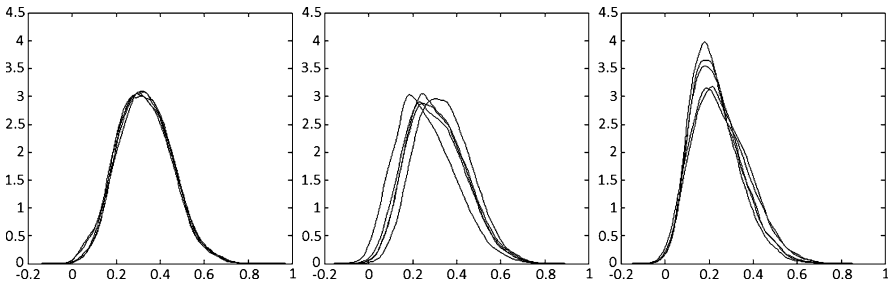


Fig. 3 The correlation distribution for Indian stock market

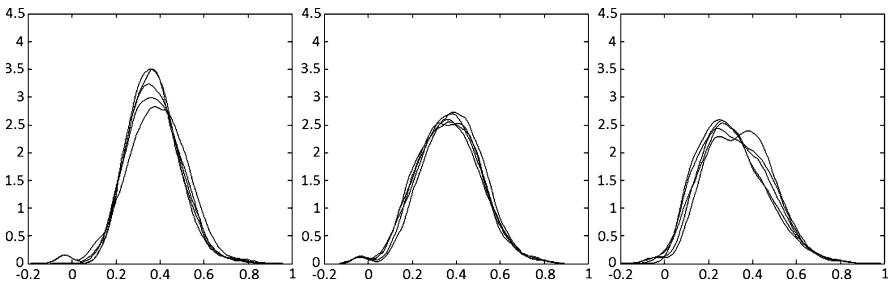


Fig. 4 The correlation distribution for Chinese stock market

Table 3 Mean correlation and standard deviation values for different periods

#	Brazil		Russia		India		China	
1	0.21	0.13	0.19	0.18	0.33	0.12	0.37	0.11
2	0.22	0.12	0.18	0.18	0.32	0.12	0.37	0.11
3	0.22	0.13	0.2	0.17	0.33	0.12	0.37	0.12
4	0.22	0.13	0.2	0.17	0.34	0.12	0.37	0.13
5	0.22	0.13	0.2	0.17	0.33	0.12	0.38	0.14
6	0.22	0.13	0.2	0.16	0.34	0.13	0.39	0.14
7	0.21	0.13	0.18	0.16	0.31	0.13	0.37	0.15
8	0.23	0.13	0.18	0.16	0.31	0.13	0.36	0.15
9	0.23	0.14	0.17	0.16	0.3	0.13	0.36	0.15
10	0.15	0.11	0.13	0.13	0.26	0.13	0.36	0.15
11	0.14	0.11	0.14	0.13	0.25	0.13	0.34	0.15
12	0.14	0.11	0.15	0.13	0.26	0.13	0.33	0.15
13	0.14	0.11	0.15	0.12	0.23	0.11	0.32	0.16
14	0.17	0.12	0.15	0.12	0.22	0.11	0.31	0.16
15	0.18	0.12	0.16	0.12	0.23	0.11	0.3	0.15
Average	0.19	0.12	0.17	0.15	0.29	0.12	0.35	0.14

shapes. The standard deviations of correlations for these countries are equal to 0.12 both but mean values are different. For Brazilian stock market, the mean value is equal to 0.19 and for Indian stock market mean value is equal to 0.29. It can be interpreted as the Indian stock market is more dependent than Brazilian stock market. The peculiarity of the Russian stock market curve is the existence of the “heavy” tail on the right side. The peculiarity of the Chinese stock market curve is the existence of the significant amount of the negative correlations.

We can also observe that the shapes of the curves are relatively stable during the first 5 periods for Brazil, Russia and India. During periods from 6 to 10 the curve’s shape changes over the time. During the periods from 11 to 15, the curve’s shape for Russian stock market is stable again. For Brazil and India, the shape for the periods from 11 to 15 is different for each period but the changes are less than during the periods from 5 to 10. For China, we can observe the different tendency. The stability of the curve for Chinese stock market takes place during the periods from 5 to 10. During the first and last 5 periods, the curve’s shape changes over time. So considering the dynamics of the correlation shape curve the Brazilian, Russian and Indian stock markets are similar to each other but the Chinese stock market is different.

## 4 Analysis of the Market Graph Models

By fixing the threshold value, we define the particular market graph. If we choose the threshold close to  $-1$ , then we will get the complete graph and the size of the maximum clique will be equal to the vertex number. If we choose the threshold close to  $1$ , then the graph will not be connected. To have accurate and comparable results of the calculations, we base our choice of the thresholds on the investigation of the Russian stock market (Vizgunov et al. 2013) [8]. The most interesting results for Russian stock market graph were obtained by choosing threshold’s values from interval  $[0.5; 0.7]$  for the maximum cliques’ calculations. To have comparable results for other BRIC countries, we use such threshold values where the density of the graphs is equal to the density of the Russian stock market graph. So for Brazilian stock market, we consider threshold values 0.44, 0.51, 0.59, for Indian stock market we consider threshold values 0.53, 0.61, 0.67, for Chinese stock market we consider threshold values 0.61, 0.7, 0.79.

For all considered threshold values we produce the stock market graphs and calculate the maximum clique. In order to find the maximum clique we use algorithm suggested by Batsyn (Batsyn et al. 2013) [9]. The results of our calculations are presented in Table 4.

The analysis of Table 4 shows that we can split our 15 periods into two groups. It is also consistent with correlation curve shape analysis. For almost all countries, stock markets the size of maximum clique grows or becomes stable during the first 9–10 periods and decreases during the last 5 periods. It holds for almost all threshold values for Brazil, Russia and India. For Russia, the tendency is clearer than for

**Table 4** Maximum clique size dynamics

#	Brazil			Russia			India			China		
1	14	7	3	20	13	7	16	8	4	10	6	3
2	13	6	4	21	12	7	17	9	4	12	7	3
3	12	9	5	20	12	8	18	10	4	13	9	4
4	12	8	5	21	13	8	19	11	5	14	11	3
5	15	9	6	20	12	8	19	11	5	14	11	5
6	14	10	5	21	13	8	21	12	5	19	12	7
7	12	9	4	21	12	7	18	11	6	17	11	6
8	16	12	8	23	12	7	20	12	6	17	11	5
9	16	12	9	24	12	7	20	10	5	17	11	5
10	11	9	7	14	9	4	15	8	5	15	11	4
11	10	7	5	12	7	3	13	6	3	13	9	5
12	10	7	4	10	6	4	12	5	3	12	7	6
13	11	7	5	10	6	3	8	4	3	13	7	6
14	12	7	6	9	6	3	6	4	3	14	7	5
15	13	8	5	14	6	3	8	4	3	13	7	4

Brazil and India. For example, the maximum clique size decreases from 24 to 14 in period 10 for threshold value 0.5. For China, the tendency is less clear. It holds only for threshold values 0.61 and 0.7 and change of the maximum clique size in periods 9–10 is relatively small.

The first group consisting of the first 9–10 periods includes the time of the global economic crises of 2008. The last 5–6 periods constitute the second group and can be considered as after crisis periods. They are characterized by less number of highly correlated stocks.

In order to calculate the structure of the maximum clique, we exclude from the consideration the stocks which are traded only during several periods. After applying this procedure we have in our model data 107 stocks for Brazil, 119 stocks for Russia, 178 stocks for India and 125 stocks for China.

The calculated values of the maximum clique size for these stocks are different from values presented in Table 4 but the difference is small and the tendency to have different behavior during first and second periods' group holds as well. In order to analyze the structure of the maximum cliques for different periods, we calculate all maximum cliques by the algorithm suggested by Batsyn et al. (2013) [9]. To understand the dynamics of the maximum cliques, we consider the union of all maximum cliques. This use of union is justified by the difficulties to choose a particular maximum clique for each period of observations. Table 5 shows the information about stocks which are elements of the union of all maximum cliques during more than 7 periods in the market graph with thresholds equal to 0.5.

We can observe that during most of the periods the maximum cliques have a large intersection for all countries. The elements of this intersection form one stable large cluster, kernel.

**Table 5** Stocks in union of maximum cliques in more than 7 periods

Country	Enterprise	#	Volume (%)
Brazil	Gerdau	15	3.12
Brazil	Metalurgica Gerdau	15	0.81
Brazil	Banco Bradesco	13	4.15
Brazil	Banco do Brasil	13	2.79
Brazil	Cyrela Brazil Realty Emprs e Parts	12	1.51
Brazil	Vale S.A.	11	16.88
Brazil	Klabin	9	0.29
Brazil	B2w—Companhia Global do Varejo	9	0.69
Brazil	Itausa—Investimentos Itau S.A.	8	2.13
Russia	Gazprom	13	28.00
Russia	Rosneft Oil Company	13	5.44
Russia	Sberbank of Russia	13	26.77
Russia	Lukoil Oil Company	13	9.74
Russia	Tatneft imeni V D Shashina	12	0.72
Russia	MTS	11	0.58
Russia	Sberbank of Russia preferred share	10	3.30
Russia	Gazprom neft	10	0.36
Russia	Surgutneftegaz	9	0.93
Russia	Tatneft imeni V D Shashina preferred share	9	0.04
Russia	NLMK	9	0.24
Russia	SeverStal	9	1.08
India	DLF Ltd.	13	1.56
India	Jaiprakash Associates Ltd.	12	0.96
India	State Bank of India	11	2.52
India	Kotak Mahindra Bank Ltd.	11	0.38
India	IDBI Bank Ltd.	11	0.44
India	Reliance Capital Ltd.	11	2.46
India	ICICI Bank Ltd.	10	1.86
India	Larsen & Toubro Ltd.	10	1.53
India	Reliance Infrastructure Ltd.	9	10.92
India	Axis Bank Ltd.	8	0.77
India	IDFC Ltd.	8	0.60
India	YES Bank Ltd.	8	0.23
China	BANK OF COMMUNICATIONS CO., LTD.	15	1.07
China	CHINA CITIC BANK CORPORATION LIMITED	15	0.55
China	CHINA MERCHANTS BANK CO., LTD.	14	3.15
China	HUA XIA BANK CO., LTD.	14	1.14
China	BANK OF BEIJING CO., LTD.	14	0.81

**Table 5** (Continued)

Country	Enterprise	#	Volume (%)
China	BANK OF NINGBO CO., LTD.	13	0.47
China	CHINA CONSTRUCTION BANK CORPORATION	12	1.15
China	INDUSTRIAL BANK CO., LTD.	12	1.82
China	BANK OF CHINA LIMITED	10	0.96
China	INDUSTRIAL AND COMMERCIAL BANK OF CHINA LIMITED	9	2.04
China	BANK OF NANJING CO., LTD.	8	0.51

The kernel stocks volume of trades is different for each country but for all country except China is significantly bigger than the average volume. Our current computational results show that for the Brazilian stock market the kernel stocks account 32 % of the total volume of the market, for the Russian stock market the kernel stocks account 77 % of the total volume of the market, for the Indian stock market the kernel stocks account 24 % of the total volume of the market. Note that the Russian stock market has two stocks (Gazprom and Sberbank of Russia) that account more than 50 % of the total volume. The Chinese stock market is different regarding this characteristic. For the Chinese stock market, the kernel stocks account 14 % of the total volume of the market which is equal to the average volume.

## 5 Conclusions

All studied characteristics can be split into two groups. All BRIC countries stock markets have similar dynamics considering the first group of the characteristics. These characteristics are as follows: trades intensity, existence of stable maximum cliques' intersection, kernel, and influence of the global economic crises 2008. The influence of the global economic crises 2008 is observed by significant change of the correlation histogram shape, mean value, standard deviation and maximum clique size for time periods including 2008 year.

At the same time we can split BRIC countries stock markets regarding second group of the characteristics. Brazil, Russia and India stock markets show the conformed changes of the correlation histogram shape and significantly bigger trades volume of kernel stocks. The Chinese stock market correlation histogram shape changes in different periods and trades volume of kernel stocks is equal to average volume.

## References

1. Boginski, V., Butenko, S., Pardalos, P.M.: On structural properties of the market graph. In: Nagurney, A. (ed.) *Innovations in Financial and Economic Networks*, pp. 29–45. Edward Elgar, Cheltenham Glos (2003)

2. Boginski, V., Butenko, S., Pardalos, P.M.: Statistical analysis of financial networks. *Comput. Stat. Data Anal.* **48**, 431–443 (2005)
3. Boginski, V., Butenko, S., Pardalos, P.M.: Mining market data: a network approach. *Comput. Oper. Res.*, 3171–3184 (2006)
4. Huang, W.-Q., Zhuang, X.-T., Shuang, Y.: A network analysis of the Chinese stock market. *Physica A* **388**, 2956–2964 (2009)
5. Jallo, D., Budai, D., Boginski, V., Goldengorin, B., Pardalos, P.M.: Network-based representation of stock market dynamics: an application to American and Swedish stock markets. In: Goldengorin, B., Kalyagin, V., Pardalos, P. (eds.) *Models, Algorithms, and Technologies for Network Analysis*. Springer Proceedings in Mathematics & Statistics, vol. 32, pp. 91–98 (2012)
6. Mantegna, R.N., Stanley, H.E.: *An Introduction to Econophysics: Correlations and Complexity in Finance*. Cambridge University Press, Cambridge (2000)
7. Salter-Townshend, M., White, A., Gollini, I., Murphy, T.: Review of statistical network analysis: models, algorithms, and software. *Stat. Anal. Data Min.* **5**(4), 243–264 (2012)
8. Vizgunov, A., Goldengorin, B., Kalyagin, V., Koldanov, A., Koldanov, P., Pardalos, P.M.: Network approach for the Russian stock market. *Comput. Manag. Sci.* (2013). doi:[10.1007/s10287-013-0165-7](https://doi.org/10.1007/s10287-013-0165-7)
9. Batsyn, M.V., Goldengorin, B.I., Maslov, E.V., Pardalos, P.M.: Improvements to MCS algorithm for the maximum clique problem. *J. Comb. Optim.* (2013). doi:[10.1007/s10878-012-9592-6](https://doi.org/10.1007/s10878-012-9592-6)



# Sensor Cover and Double Partition

Lidong Wu, Weili Wu, Zaixin Lu, Yuqing Zhu, and Ding-Zhu Du

**Abstract** The minimum sensor cover is an important optimization problem in wireless sensor networks, which can be seen as a special case of the minimum set cover problem. The minimum set cover problem is a well-known problem in combinatorial optimization, which has no polynomial-time  $(\rho \ln \delta)$ -approximation for  $0 < \rho < 1$  unless  $NP \subseteq DTIME(n^{O(\log \log n)})$  where  $\delta$  is the maximum cardinality of a subset in the input collection. However, the minimum sensor cover problem has polynomial-time constant-approximations because this special case has a geometric structure. The design technique, called *partition*, is employed to take the advantage of this geometric structure. Especially, the *double partition* plays an important role. In this article, we would like to give an exploratory essay for the double partition technique together with research progress on approximations for the minimum sensor cover problem.

**Keywords** Minimum set cover · Minimum sensor cover · Wireless sensor network · Double partition

## 1 Introduction

In this article, we are studying an optimization problem in wireless sensor networks as follows.

---

L. Wu (✉) · W. Wu · Z. Lu · Y. Zhu · D.-Z. Du  
Department of Computer Science, University of Texas at Dallas, Richardson TX 75080, USA  
e-mail: [lidong.wu@utdallas.edu](mailto:lidong.wu@utdallas.edu)

W. Wu  
e-mail: [weiliwu@utdallas.edu](mailto:weiliwu@utdallas.edu)

Z. Lu  
e-mail: [zaixinlu@utdallas.edu](mailto:zaixinlu@utdallas.edu)

Y. Zhu  
e-mail: [yuqing.zhu@utdallas.edu](mailto:yuqing.zhu@utdallas.edu)

D.-Z. Du  
e-mail: [dzdu@utdallas.edu](mailto:dzdu@utdallas.edu)

Consider a set  $T$  of  $n$  (points) targets and a set  $S$  of  $m$  sensors in the Euclidean plane. Suppose all sensors have the same sensing radius, i.e., each sensor has a sensing disk with radius  $R_s$ . For simplicity, assume  $R_s = 1$ . To save energy, we may find a minimum subset of sensors monitoring all targets and then put other sensors in sleeping phase. This problem is called the minimum sensor cover problem, which can also be described in the following mathematical way.

**MINSC:** Given a set  $T$  of targets and a set  $S$  of disks with radius of one in the Euclidean plane, find a minimum cardinality subset  $S'$  of  $S$  to cover all targets.

This problem is a special case of the minimum set cover problem which is very well-known in combinatorial optimization as follows.

**SET-COVER:** Given a collection  $\mathcal{C}$  of subsets of a finite set  $X$ , find a minimum-cardinality subcollection  $\mathcal{C}'$  of  $\mathcal{C}$  to cover all elements in  $X$ , i.e.,  $\bigcup_{A \in \mathcal{C}'} A = X$ .

It has been known that SET-COVER has a polynomial-time greedy  $(1 + \ln \delta)$ -approximation where  $\delta = \max_{A \in \mathcal{C}} |A|$ . It is also known that SET-COVER problem has no polynomial-time  $(\rho \ln \delta)$ -approximation for  $0 < \rho < 1$  unless  $NP \subseteq DTIME(n^{O(\log \log n)})$ . However, MIN-SC has polynomial-time constant approximations because MIN-SC has a geometric structure; and a design technique, called *partition*, can be employed to take the advantage of this geometric structure to establish a better approximation. Especially, the *double partition* plays an important role. In this article, we would like to give an exploratory essay for the double partition technique together with research progress on approximations for MINSC.

## 2 A Special Case

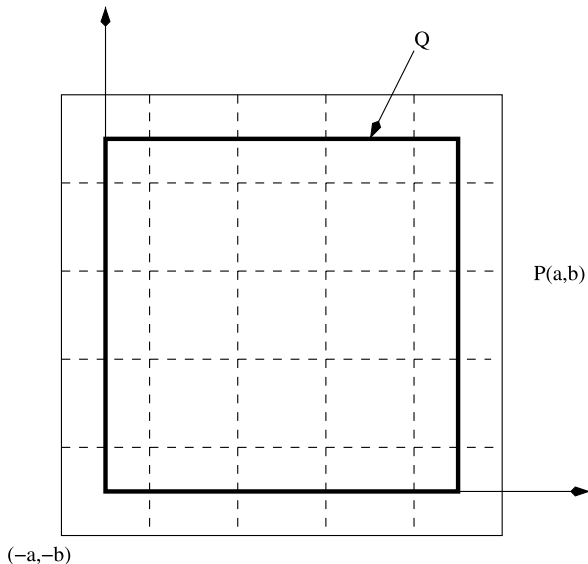
Let us first consider a special case that every sensor is a target and nothing else is a target. Consider all sensors as vertices. For any two vertices with distance at most one, connect them with an edge. We would obtain a unit disk graph  $G$ . Since the sensor set and the target set are identical to the vertex set, a sensor cover is a dominating set, i.e., a vertex subset such that every vertex is either in the subset or adjacent to a vertex in the subset. Therefore, this special case is equivalent to the minimum dominating set problem in unit disk graphs as follows.

**MINDS-IN-UDG:** Given a unit disk graph  $G = (V, E)$ , find the minimum-cardinality dominating set of  $G$ .

With partition technique, we can design a PTAS for MINDS-IN-UDG, that is, for any  $\varepsilon > 0$ , design a polynomial-time algorithm which produces an approximation solution with objective function value within a factor of  $(1 + \varepsilon)$  from optimal.

To do so, we first put the input graph  $G$  in a square  $Q$  with side length  $p$ . Put the square  $Q$  in a coordinate system with the origin  $(0, 0)$  at the left lower vertex of the square. Construct a  $qm\mu \times qm\mu$  grid with  $q^2$  blocks; each block is a square of side length  $m\mu$  where  $\mu = \sqrt{2}/2$  and  $m$  and  $q$  are integers such that  $q \geq \lceil \frac{p}{m\mu} \rceil + 1$ .

The grid is used for partitioning the square  $Q$ . When we put the left lower corner of the grid at point  $(-a, -b)$ , the resulting partition is denoted by  $P(a, b)$  (Fig. 1).

**Fig. 1** Partition  $P(a, b)$ 

To make all blocks in a partition disjoint, let each block does not contain upper boundary and right boundary, that is, each block is obtained from a close square by removing the upper boundary and the right boundary.

Consider a block  $B$ . Let  $V(B)$  be the subset of all vertices in  $B$  and  $W(B)$  the subset of all vertices each adjacent to a vertex in  $B$ . We consider a problem on  $B$  as follows:

**MINDS( $B$ ):** Given unit disk graph  $G$  and a block  $B$ , find the minimum-cardinality subset of  $W(B)$  to dominate  $V(B)$ .

We first show the following.

**Lemma 1** *MINDS( $B$ ) can be solved in time  $|W(B)|^{O(m^2)}$ .*

*Proof* The block  $B$  can be partitioned into  $m^2$  squares with side length  $\mu$ . Such squares are called cells. For each cell, if it contains a vertex  $x$ , then all vertices in this cell can be dominated by  $x$ . This means that all vertices lying in the block  $B$  can be dominated by at most  $m^2$  vertices. Therefore, an optimal solution for MINDS( $B$ ) contains at most  $m^2$  vertices. Note that  $W(B)$  contains at most

$$\binom{|W(B)|}{1} + \binom{|W(B)|}{1} + \cdots + \binom{|W(B)|}{m^2} = |W(B)|^{O(m^2)}$$

nonempty subsets of size at most  $m^2$ . For each such subset  $U$ , determining whether  $U$  dominates  $V(B)$  can be done in time  $O(|V(B)|)$ . Therefore, an optimal solution can be found in time  $|W(B)|^{O(m^2)}$  by a brute-force search among above vertex subsets from small size to large size.  $\square$

Let  $Opt(B)$  be an optimal solution for  $MINDS(B)$ . Let

$$A(a, b) = \bigcup_{B \in P(a, b)} Opt(B).$$

Let us compare  $A(a, b)$  with an optimal solution  $Opt_{\text{mds}}$  for  $MINDS\text{-IN-UDG}$ . Denote

$$Opt_{\text{mds}}(B) = \{v \in Opt_{\text{mds}} \mid v \text{ is adjacent to a vertex in } V(B)\}.$$

Then  $Opt_{\text{mds}}(B)$  is a feasible solution for  $MINDS(B)$ . Therefore,

$$\begin{aligned} |A(a, b)| &\leq \sum_{B \in P(a, b)} |Opt(B)| \\ &\leq \sum_{B \in P(a, b)} |Opt_{\text{mds}}(B)| \\ &= |Opt_{\text{mds}}| + n_2(a, b) + 2n_3(a, b) + 3n_4(a, b) \end{aligned}$$

where  $n_i(a, b)$  is the number of vertices in  $Opt_{\text{mds}}$  each of which has neighbors, including itself, in exactly  $i$  blocks.

Clearly, the best partition  $P(a, b)$  is to minimize  $n_2(a, b) + 2n_3(a, b) + 3n_4(a, b)$ . However, it is not easy to be found because it requires to compute  $Opt_{\text{mds}}$  which is unlikely to be done in polynomial-time. Instead, we may do the following.

- (1) Find a subset  $D$  of  $(a, b)$ 's such that for  $(a, b)$  over this subset  $D$ , the average value of  $n_2(a, b) + 2n_3(a, b) + 3n_4(a, b)$  can be established easily.
- (2) Compute the minimum value  $|A(a, b)|$  for  $(a, b)$  over  $D$ . This minimum value would have  $Opt_{\text{mds}}$  plus the above average value as an upper bound so that a theoretical performance ratio can be established.

Motivated from this idea, we obtain the following algorithm.

### Algorithm 1

**input** a unit disk graph  $G = (V, E)$ ;

$min \leftarrow +\infty$ ;

$A \leftarrow \emptyset$ ;

**for**  $i = 0$  **to**  $m/3 - 1$  **do**

**if**  $|A(3i\mu, 3i\mu)| < min$

**then**  $min \leftarrow |A(3i\mu, 3i\mu)|$

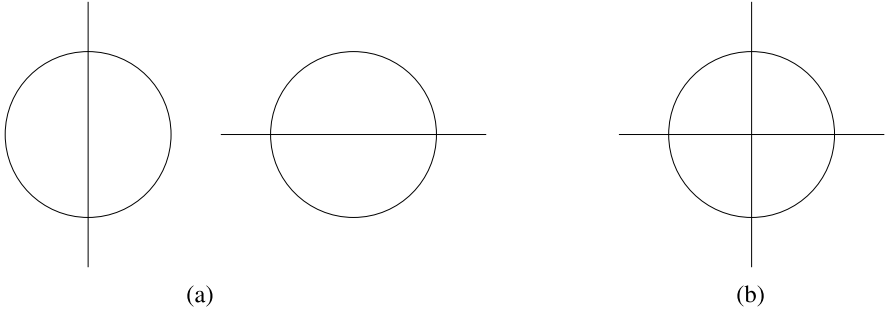
$A \leftarrow A(3i\mu, 3i\mu)$ ;

**output**  $A_{\text{out}}(1) = A$ .

Indeed, we have

**Lemma 2** Choose  $m$  to be divisible by 3. Then

$$\sum_{i=0}^{m/3-1} [n_2(3i\mu, 3i\mu) + 2n_3(3i\mu, 3i\mu) + 3n_4(3i\mu, 3i\mu)] \leq 3 \cdot |Opt_{\text{mds}}|. \quad (1)$$



**Fig. 2** A sensing disk intersects cutlines

*Proof* Recall that the sensing disk of a vertex  $v$  (as a sensor) is the disk with center  $v$  and radius one. If a vertex in  $Opt_{\text{mds}}$  has neighbors, including itself, in exactly 2 blocks in partition  $P(a, b)$ , then its sensing disk would intersect either a horizontal cutline or a vertical cutline in  $P(a, b)$ , see Fig. 2(a). If a vertex in  $Opt_{\text{mds}}$  has neighbors, including itself, in at least 3 blocks in partition  $P(a, b)$ , then its sensing disk would intersect both horizontal and vertical cutlines in  $P(a, b)$  see Fig. 2(b).

Now, any two horizontal cutlines in  $P(0, 0) \cup P(3\mu, 3\mu) \cup \dots \cup P((m-3)\mu, (m-3)\mu)$  have distance at least  $3\mu > 2$ . Therefore, the sensing disk of a vertex can intersect at most one horizontal cutline, and similarly at most one vertical cutline in  $P(0, 0) \cup P(3\mu, 3\mu) \cup \dots \cup P((m-3)\mu, (m-3)\mu)$ . This means that each vertex in  $Opt_{\text{mds}}$  can contribute at most 3 to the left side of (1). Hence (1) holds.  $\square$

By Lemma 2, the size of output of Algorithm 1 has the following upper bound:

$$\begin{aligned}
 |A_{\text{out}}(1)| &= \min_{0 \leq i \leq m/3-1} |A(3i\mu, 3i\mu)| \\
 &\leq \frac{1}{m/3} \sum_{i=0}^{m/3-1} |A(3i\mu, 3i\mu)| \\
 &\leq \left(1 + \frac{9}{m}\right) |Opt_{\text{mds}}|.
 \end{aligned}$$

Choose  $m \geq 9/\varepsilon$ . Then we have

$$|A_{\text{out}}(1)| \leq (1 + \varepsilon) |Opt_{\text{mds}}|.$$

This means that the following holds.

**Theorem 1** MINDS-IN-UDG has a PTAS.

### 3 Extension to General Case

In this section, we examine where in last section can be extended to the general case and where cannot.

First, draw a square  $Q$  with side length  $p$  to contain all input sensors and all input targets. Put the square  $Q$  in a coordinate system with the origin  $(0, 0)$  at the left lower vertex of the square. Construct a  $qm\mu \times qm\mu$  grid with  $q^2$  blocks; each block is a square of side length  $m\mu$  where  $\mu = \sqrt{2}/2$  and  $m$  and  $q$  are integers such that  $q \geq \lceil \frac{p}{m\mu} \rceil + 1$ .

The grid is used for partitioning the square  $Q$ . When we put the left lower corner of the grid at point  $(-a, -b)$ , the resulting partition is denoted by  $P(a, b)$  (Fig. 1). To make all blocks in a partition disjoint, let each block does not contain upper boundary and right boundary, that is, each block is obtained from a close square by removing the upper boundary and the right boundary.

Consider a block  $B$ . Let  $T(B)$  be the subset of all targets lying in  $B$  and  $S(B)$  the subset of all sensors each covering at least one target in  $T(B)$ . We consider a problem on  $B$  as follows:

**MINSC( $B$ ):** Given a sensor set  $S$ , a target set  $T$  and a block  $B$ , find the minimum-cardinality subset of  $S(B)$  to cover  $T(B)$ .

Now, Lemma 1 cannot be extended to the general case, that is, a polynomial-time solution has not been found for MINSC( $B$ ). The difficulty is that the size of an optimal solution for MINSC( $B$ ) cannot be easily upper-bounded by a constant. In fact, a square with side length  $\mu$  containing a target does not imply the square containing a sensor and hence no constant upper bound can be established for the number of sensors covering all targets in the square.

Let us assume that MINSC( $B$ ) has a polynomial-time  $\rho$ -approximation  $A_\rho(B)$  and continue our examination.

Let  $A_\rho(a, b) = \bigcup_{B \in P(a, b)} A_\rho(B)$ . Let us compare  $A(a, b)$  with an optimal solution  $Opt_{\text{msc}}$  for MINSC. Denote

$$Opt_{\text{msc}}(B) = \{v \in Opt_{\text{msc}} \mid v \text{ covers a target in } T(B)\}.$$

Then  $Opt_{\text{msc}}(B)$  is a feasible solution for MINSC( $B$ ). Therefore,

$$\begin{aligned} |A_\rho(a, b)| &\leq \sum_{B \in P(a, b)} |A_\rho(B)| \\ &\leq \sum_{B \in P(a, b)} \rho |Opt_{\text{msc}}(B)| \\ &= \rho (|Opt_{\text{msc}}| + n_2(a, b) + 2n_3(a, b) + 3n_4(a, b)) \end{aligned}$$

where  $n_i(a, b)$  is the number of sensors in  $Opt_{\text{msc}}$  each of which covers targets in exactly  $i$  blocks.

Clearly, the best partition  $P(a, b)$  is to minimize  $n_2(a, b) + 2n_3(a, b) + 3n_4(a, b)$ . However, it is not easy to be found because it requires to compute  $Opt_{\text{msc}}$  which is unlikely to be done in polynomial-time. Instead, we may do the following.

- (1) Find a subset  $D$  of  $(a, b)$ 's such that for  $(a, b)$  over this subset  $D$ , the average value of  $n_2(a, b) + 2n_3(a, b) + 3n_4(a, b)$  can be established easily.
- (2) Compute the minimum value  $|A_\rho(a, b)|$  for  $(a, b)$  over  $D$ . This minimum value would have  $Opt_{\text{mds}}$  plus the above average value as an upper bound so that a theoretical performance ratio can be established.

Motivated from this idea and the fact that Lemma 2 still holds, we obtain the following algorithm.

### Algorithm 2

**input** a sensor set  $S$  and a target set  $T$ ;

$min \leftarrow +\infty$ ;

$A \leftarrow \emptyset$ ;

**for**  $i = 0$  **to**  $m/3 - 1$  **do**

**if**  $|A_\rho(3i\mu, 3i\mu)| < min$

**then**  $min \leftarrow |A_\rho(3i\mu, 3i\mu)|$

$A \leftarrow A_\rho(3i\mu, 3i\mu)$ ;

**output**  $A_{\text{out}}(2) = A$ .

By Lemma 2, the size of output of Algorithm 2 has the following upper bound:

$$\begin{aligned}
 |A_{\text{out}}(2)| &= \min_{0 \leq i \leq m/3-1} |A_\rho(3i\mu, 3i\mu)| \\
 &\leq \frac{1}{m/3} \sum_{i=0}^{m/3-1} |A_\rho(3i\mu, 3i\mu)| \\
 &\leq \rho \left(1 + \frac{9}{m}\right) |Opt_{\text{msc}}|.
 \end{aligned}$$

Choose  $m \geq \rho \cdot 9/\varepsilon$ . Then we have

$$|A_{\text{out}}(2)| \leq (\rho + \varepsilon) |Opt_{\text{msc}}|.$$

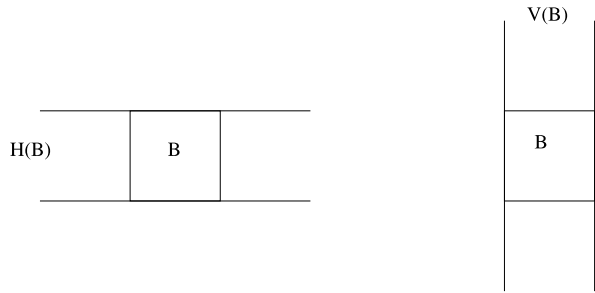
This means that the following holds.

**Theorem 2** *Suppose that for  $m \geq 3$  with  $m$  divisible by 3,  $\text{MINSC}(B)$  has a polynomial-time  $\rho$ -approximation where  $\rho$  is a constant. Then  $\text{MINSC}$  has a polynomial-time  $\rho(1 + \frac{9}{m})$ -approximation.*

**Corollary 1** *Suppose that for any  $m \geq 3$  with  $m$  divisible by 3,  $\text{MINSC}(B)$  has a polynomial-time  $\rho$ -approximation where  $\rho$  is a constant. Then  $\text{MINSC}$  has a polynomial-time  $(\rho + \varepsilon)$ -approximation for any  $\varepsilon > 0$ .*

In above result, it seems that  $\text{MINSC}(B)$  for  $m = 1$  is useless. However, it is the most important case. We will discuss it in next section.

**Fig. 3** Strips  $H(B)$  and  $V(B)$



#### 4 MINSC( $B$ ) for $m = 1$

Ambühl et al. [1] showed that MINSC( $B$ ) for a square with side length smaller than  $\mu$  has a polynomial-time 2-approximation. Huang et al. [11, 14] showed the following with a different approach.

**Lemma 3** (Huang et al. [11, 14]) *MINSC( $B$ ) for  $m = 1$  has a polynomial-time 2-approximation.*

Both approaches are based on a polynomial-time solution for the following problem.

**MINSC-STRIP:** Given a set  $T$  of targets lying inside a strip and a set  $S$  of sensors lying outside the strip, and for each sensor  $s \in S$ , given a nonnegative weight  $c(s)$ , find the minimum total weight subset of sensors to cover all targets.

**Theorem 3** (Ambühl et al. [1]) *MINSC-STRIP can be solved in time  $O(|S|^4 \cdot |T|)$ .*

Since  $m = 1$ , a block  $B$  is a square with side length  $\mu$ . If  $T(B) \neq \emptyset$  and  $S \cap B \neq \emptyset$ , then any sensor in  $S \cap B$  can cover all targets in  $T(B)$  and hence forms an optimal solution for MINSC( $B$ ). If  $T(B) \neq \emptyset$  and  $S \cap B = \emptyset$ , then Huang et al. [11, 14] showed a decomposition as follows.

**Lemma 4** (Huang et al. [11, 14]) *The optimal solution of MINSC( $B$ ) for  $m = 1$  and  $T(B) \neq \emptyset$  either consists of a sensor in  $S \cap B$  or can be decomposed into not-necessarily disjoint two parts  $Opt_1(B)$  and  $Opt_2(B)$  satisfying the following conditions:*

- (1) *All sensors in  $Opt_1(B)$  lie outside the strip  $H(B)$  obtained from  $B$  by extending two horizontal sides, see Fig. 3.*
- (2) *All sensors in  $Opt_2(B)$  lie outside the strip  $V(B)$  obtained from  $B$  by extending two vertical sides, see Fig. 3.*
- (3)  *$T(B)$  has a partition  $(T_1(B), T_2(B))$ , which can be determined by at most four targets in  $T(B)$ , such that  $T_1(B)$  can be covered by  $Opt_1(B)$  and  $T_2(B)$  can be covered by  $Opt_2(B)$ .*



By Lemma 4 and Theorem 3, a polynomial-time 2-approximation for  $\text{MINSC}(B)$  with  $m = 1$  can be computed as follows.

**Algorithm 3** (2-Approximation for  $\text{MINSC}(B)$  with  $m = 1$ )

**input** a sensor set  $S$ , a target set  $T$  and a block  $B$  with side length  $\mu$ ;  
**if**  $T(B) = \emptyset$   
    **then**  $A_{\text{out}}(3) \leftarrow \emptyset$   
**else if**  $S(B) \cap B \neq \emptyset$   
    **then** choose  $s \in S(B) \cap B$  and set  $A_{\text{out}}(3) \leftarrow \{s\}$   
    **else begin**  
        let  $S_1(B) = S(B) \setminus H(B)$  and  $S_2(B) = S(B) \setminus V(B)$ ;  
        **for** each subset  $T'$  of at most four targets **do begin**  
            use  $T'$  to determine a partition of  $T(B)$ ,  $(T_1(B), T_2(B))$   
            with approach in Huang et al. [11, 14];  
            **if**  $S_1(B)$  can cover  $T_1(B)$  and  $S_2(B)$  can cover  $T_2(B)$   
                **then** find the minimum subset  $U_1$  of  $T_1(B)$  to cover  $T_1(B)$   
                    and the minimum subset  $U_2$  of  $T_2(B)$  to cover  $T_2(B)$ ,  
                    and set  $A(T') \leftarrow U_1 \cup U_2$   
                **else**  $A(T') \leftarrow S(B)$ ;  
            **end-for**;  
            set  $T'_{\min} = \text{argmin}_{T'} |A(T')|$ ;  
             $A_{\text{out}}(3) \leftarrow A(T'_{\min})$ ;  
        **end-else**;  
**output**  $A_{\text{out}}(3)$ .

This algorithm runs in time  $O(|T(B)|^5 \cdot |S(B)|^4)$ .

Let us first use this 2-approximation solution directly to design a polynomial-time 28-approximation for  $\text{MINSC}$ .

**Algorithm 4** (28-Approximation for  $\text{MINSC}$ )

**input** a sensor set  $S$  and a target set  $T$ ;  
**for** each block  $B$  in  $P(0, 0)$  **do**  
    compute 2-approximation  $A_2(B)$ ;  
**output**  $A_{\text{out}}(4) = \bigcup_{B \in P(0, 0)} A_2(B)$ .

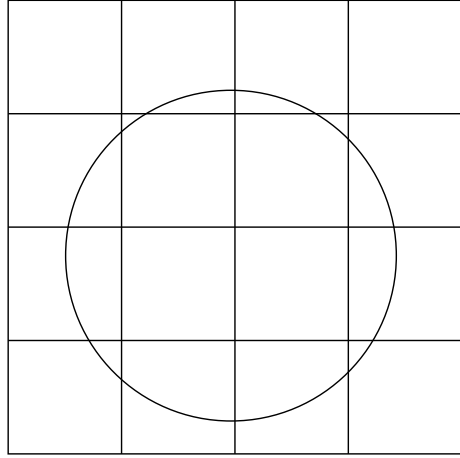
To analyze Algorithm 4, we first note that each sensor  $s \in S$  can be in  $S(B)$  for at most 14 blocks  $B$ . Indeed, the sensing disk of  $s$  lies in four vertical strips and four horizontal strips, see Fig. 4; the intersection of four horizontal strips and four vertical strips contains 16 blocks. However, the sensing disk of  $s$  can intersect only one of the two blocks at opposite corners on any diagonal.

**Theorem 4** Algorithm 4 is a polynomial-time 28-approximation for  $\text{MINSC}$ .

*Proof* Let  $\text{Opt}_{\text{msc}}$  be an optimal solution for  $\text{MINSC}$  and for any block  $B$ , let

$$\text{Opt}_{\text{msc}}(B) = \{s \in \text{Opt}_{\text{msc}} \mid s \text{ covers a target in } B\}.$$

**Fig. 4** A sensing disk can intersect at most 14 blocks



Then  $Opt_{\text{msc}}(B)$  is a feasible solution for  $\text{MINSC}(B)$ . Since each sensor can be in  $S(B)$  for at most 14 blocks  $B$ , we have

$$\begin{aligned}
 |A_{\text{out}}(4)| &\leq \sum_{B \in P(0,0)} |A_2(B)| \\
 &\leq 2 \sum_{B \in P(0,0)} |Opt_{\text{msc}}(B)| \\
 &\leq 28 |Opt_{\text{msc}}|. \quad \square
 \end{aligned}$$

## 5 $\text{MINSC}(B)$ for $m \geq 2$

In literature, the best known result for  $\text{MINSC}(B)$  with  $m \geq 2$  is as follows.

**Lemma 5** (Willson et al. [17]) *For any  $m \geq 2$ ,  $\text{MINSC}(B)$  has a polynomial-time 3.63-approximation.*

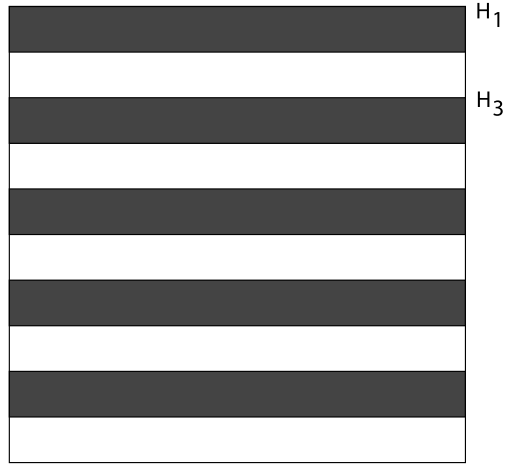
This result is obtained based on partition of block  $B$  into  $m^2$  cells; each cell is a square with side length  $\mu$ , see Fig. 5. Those  $m^2$  cells form  $m$  horizontal strips, denoted by  $H_1, \dots, H_m$ , and  $m$  vertical strips, denoted by  $V_1, \dots, V_m$ . Consider the following problem.

**MINSC-MULTISTRIP:** Given a sensor set  $S(B)$  and a target set  $T(B)$ , find the minimum subset of  $S(B)$  for properly-covering all targets lying in  $H_1 \cup H_3 \cup \dots \cup H_{2\lceil m/2 \rceil - 1}$  where a sensor  $s$  is said to properly-cover a target  $t$  if  $s$  and  $t$  do not lie in the same horizontal strip.

Erlebach and Mihalak [10] showed the following.

**Theorem 5** (Erlebach and Mihalak [10])  *$\text{MINSC-MULTISTRIP}$  is polynomial-time solvable.*

**Fig. 5** A horizontal multistrip



By Theorem 5, we can design a polynomial-time 4-approximation for  $\text{MINSC}(B)$  as follows.

**Algorithm 5** (4-Approximation for  $\text{MINSC}(B)$ )

Initially, input a sensor set  $S$ , a target set  $T$  and a block  $B$ . Divide block  $B$  into  $m^2$  cells which form a collection  $\mathcal{C}$ .

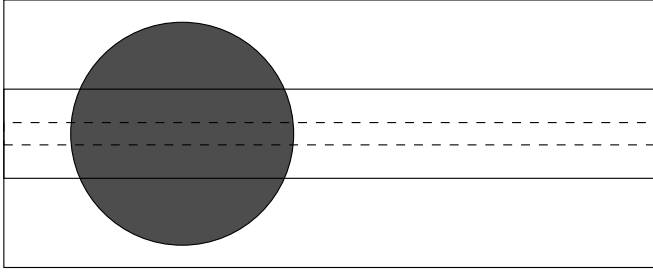
**Step 1.** For each subcollection  $\mathcal{C}'$  of  $\mathcal{C}$ , if for every  $C \in \mathcal{C}'$ ,  $C \cap T = \emptyset$  or  $S \cap C \neq \emptyset$ , then choose  $s_C \in S \cap C$  for every  $C \in \mathcal{C}'$  with  $T \cap C \neq \emptyset$ , delete from  $T$  all targets covered by  $s_C$  (the remaining set is still denoted by  $T$ ) and go to Step 2; otherwise set  $A(\mathcal{C}') = \text{nil}$  (note: *nil* means undefined.)

**Step 2.** Choose a subset  $T'$  of  $T$  such that for every cell  $C \in \mathcal{C} - \mathcal{C}'$ ,  $|T'(C)| \leq 4$ . Partition every  $T(C)$  into two parts  $T_1(C)$  and  $T_2(C)$  (based on  $T'(C)$  as in Lemma 4) and go to Step 3.

**Step 3.** If for every cell  $C \in \mathcal{C} - \mathcal{C}'$ ,  $S_1(C) = S(C) \setminus H(C)$  can cover  $T_1(C)$  and  $S_2 = S(C) \setminus V_2(C)$  can cover  $T_2(C)$ , then go to Step 4; otherwise, set  $A(\mathcal{C}', T') = \text{nil}$  and if all choices of  $T'$  have been considered, then go to Step 5, else go back Step 2.

**Step 4.** Solve  $\text{MINSC-MULTISTRIP}$  on the following four input pairs:

$$\begin{aligned} & \left( \bigcup_{C \in H_1 \cup H_3 \cup \dots} S_1(C), \bigcup_{C \in H_1 \cup H_3 \cup \dots} T_1(C) \right) \\ & \left( \bigcup_{C \in H_2 \cup H_4 \cup \dots} S_1(C), \bigcup_{C \in H_2 \cup H_4 \cup \dots} T_1(C) \right) \\ & \left( \bigcup_{C \in V_1 \cup V_3 \cup \dots} S_2(C), \bigcup_{C \in V_1 \cup V_3 \cup \dots} T_2(C) \right) \end{aligned}$$



**Fig. 6** A sensor involves only one problem on horizontal multistrip

$$\left( \bigcup_{C \in V_2 \cup V_4 \cup \dots} S_2(C), \bigcup_{C \in V_2 \cup V_4 \cup \dots} T_2(C) \right).$$

Let  $A(\mathcal{C}', T')$  be the union of four optimal solutions. If all choices of  $T'$  have been considered, then go to Step 5; else, go to Step 2.

**Step 5.** Let  $|A(\mathcal{C}', T'_{\min})| = \min_{T'} |A(\mathcal{C}', T')|$  for  $T'$  over all choices (note: define  $|nil| = 0$ ). Set

$$A(\mathcal{C}') = A(\mathcal{C}', T'_{\min}) \cup \{s_C \mid C \in \mathcal{C}' \text{ with } T \cap C \neq \emptyset\}.$$

Go to Step 6.

**Step 6.** Output  $A_{\text{out}}(5) = A(\mathcal{C}'_{\min})$  where  $|A(\mathcal{C}'_{\min})| = \min_{\mathcal{C}' \subseteq \mathcal{C}} |A(\mathcal{C}')|$ .

To see  $|A_{\text{out}}(5)| \leq 4|Opt_{\text{msc}}|$ , it suffices to see that in Step 4, there are totally 4 polynomial-time solvable problems and hence each sensor can involve at most four of them.

To improve Algorithm 5 to a 3.63-approximation, the key observation is that in some position, a sensor may involve only two or three polynomial-time solvable problems in Step 4, see Fig. 6. If we randomly deploy a sensor  $s$  into  $B$ , then it is not hard to calculate that the average number of polynomial-time solvable problems in Step 4 that  $s$  involves is

$$2 \cdot \frac{2 \cdot 2 \cdot (1 - \mu) + (3\mu - 2)}{\mu} = 4\sqrt{2} - 2 < 3.63.$$

This is because that for each horizontal strip, the area where sensor  $s$  involves only one problem on horizontal multistrip is a horizontal strip with width  $3\mu - 2$  and the rest area consists of two horizontal strips with width  $1 - \mu$ . For each vertical strip, the similar thing holds.

How to turn this number to be a performance ratio? The technique is shifting of partition of block  $B$ .

### Algorithm 6

Let  $P$  be the set of points evenly distributed in square  $[-1, 0] \times [-1, 0]$ . Shift the partition of block  $B$  and at every position in  $P$ , compute an approximation solution  $A_{\text{out}}(5)$  and let  $A_{\text{out}}(6)$  be the one with minimum cardinality.

Note that shifting of partition is equivalent to moving a sensor. Therefore, when we calculate the average of  $|A_{\text{out}}(5)|$ , each sensor would involve at most  $3.63 - \varepsilon$  problems for a sufficiently small  $\varepsilon > 0$  as  $P$  is large enough. Therefore, choose  $P$  properly,  $A_{\text{out}}(6)$  can be a polynomial-time  $(3.63 - \varepsilon)$ -approximation for  $\text{MINS}(B)$ .

By Theorem 3, we obtain

**Theorem 6** *MINS has a polynomial-time 3.63-approximation.*

## 6 Weighted Sensor Cover

All results in Sects. 3–5 can be extended to the weighted case as follows.

**MINWSC:** Given a set  $T$  of targets and a set  $S$  of sensors, with nonnegative weight  $c : S \rightarrow \mathbb{R}^+$ , in the Euclidean plane, find a minimum weight subset  $S'$  of  $S$  to cover all targets.

However, the result in Sect. 2 cannot be extended to the weighted case as follows.

**MINWDS-IN-UDG:** Given a unit disk graph  $G = (V, E)$  with nonnegative weight  $c : V \rightarrow \mathbb{R}^+$ , find the minimum-weight dominating set of  $G$ .

Indeed, it was open for many years whether MINWDS-IN-UDG has a polynomial-time constant-approximation. Ambühl et al. [1] solved this open problem by giving a polynomial-time 72-approximation. Huang et al. [11, 14] designed a polynomial-time  $(6 + \varepsilon)$ -approximation with a new technique, called *double partition*. Also with double partition, the bound  $(6 + \varepsilon)$  is improved to  $(5 + \varepsilon)$  by Dai and Yu [5], to  $(4 + \varepsilon)$  by Zou et al. [23] and independently by Erlebach and Mihalák [10], to 3.63 by Willson et al. [17].

The weighted version of Theorem 6 has an important consequence in [17]. Consider the maximum lifetime coverage problem [4] as follows.

**MAX-LIFETIME-COVERAGE:** Given a set  $T$  of targets and a set  $S$  of sensors with unit lifetime in the Euclidean plane, find a active/sleeping schedule of sensors to maximize the lifetime of coverage where the coverage is alive if all targets are covered by active sensors.

Garg and Könemann [12] showed the following.

**Lemma 6** (Garg and Könemann [12]) *If MINWSC has a polynomial-time  $\rho$ -approximation, then MAX-LIFETIME-COVERAGE has a polynomial-time  $(\rho + \varepsilon)$ -approximation for any  $\varepsilon > 0$ .*

With this lemma, Ding et al. [6] gave the first constant-approximation for MAX-LIFETIME-COVERAGE. By the weighted version of Theorem 6 with a little adjustment on 3.63 (i.e., replaced 3.63 by  $3.63 - \varepsilon$  for a very small  $\varepsilon > 0$ ), a better result can be obtained as follows.

**Theorem 7** (Willson et al. [17]) *MAX-LIFETIME-COVERAGE has a polynomial-time 3.63-approximation.*

## 7 Discussion

Wireless sensor networks have a lot of applications in the real world [3, 15, 18, 19]. In this article, we discussed several important optimization problems in wireless sensor networks and an important technique, partition for design of approximation algorithms. This technique was initiated by Baker [2] and Hochbaum and Maass [13]. To obtain 3.63-approximation for MINSC, the partition is used twice, the first one in Sect. 3 and the second one in Sect. 5. Therefore, we call this variation of partition as the double partition. More variations of partition can be found in [7, 8].

One of our purposes of this article is to see what problems are still open and how to work on those problems. Indeed, from study in previous sections, we have identified two interesting open problems as follows.

**Open Problem 1** *Does MINWSC( $B$ ) problem for  $m = 2$  have a polynomial-time 2-approximation?*

Indeed, from solving this problem, we may find some hint to solve the general case of the problem:

**Open Problem 2** *Does MINWSC( $B$ ) problem for  $m \geq 2$  have a polynomial-time 2-approximation?*

Along this direction, we may improve approximation solutions for MINSC and MAX-LIFETIME-COVERAGE.

Getting various connectivities (such as connectivity [20–22], weak-connectivity [9] or 2-connectivity [16]) involved, the sensor cover problem would have many variations. Those variations can give another source of open problems for further research.

**Acknowledgements** This work was supported in part by National Science Foundation of USA under grants CNS101630 and CCF0829993.

## References

1. Ambühl, C., Erlebach, T., Mihalák, M., Nunkesser, M.: Constant-approximation for minimum-weight (connected) dominating sets in unit disk graphs. In: Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX 2006). LNCS, vol. 4110, pp. 3–14. Springer, Berlin (2006)
2. Baker, B.S.: Approximation algorithms for NP-complete problems on planar graphs. In: Proc. FOCS, pp. 265–273 (1983)
3. Boginski, V.L., Commander, C.W., Pardalos, P.M., Ye, Y. (eds.): Sensors: Theory, Algorithms, and Applications. Springer, Berlin (2012)
4. Cardei, M., Thai, M., Li, Y., Wu, W.: Energy-efficient target coverage in wireless sensor networks. In: IEEE INFOCOM, pp. 1976–1984 (2005)
5. Dai, D., Yu, C.: A  $(5 + \epsilon)$ -approximation algorithm for minimum weighted dominating set in unit disk graph. Theor. Comput. Sci. **410**, 756–765 (2009)

6. Ding, L., Wu, W., Willson, J.K., Wu, L., Lu, Z., Lee, W.: Constant-approximation for target coverage problem in wireless sensor networks. In: Proc. of the 31st Annual Joint Conf. of IEEE Communication and Computer Society (INFOCOM) (2012)
7. Du, D.-Z., Ko, K.-I., Hu, X.: Design and Analysis of Approximation Algorithms, pp. 142–157. Springer, Berlin (2012)
8. Du, D.-Z., Wan, P.: Connected Dominating Set: Theory and Applications. Springer, Berlin (2012)
9. Du, H., Wu, W., Shan, S., Kim, D., Lee, W.: Constructing weakly connected dominating set for secure clustering in distributed sensor network. *J. Comb. Optim.* **23**, 301–307 (2012)
10. Erlebach, T., Mihalak, M.: A  $(4 + \varepsilon)$ -approximation for the minimum-weight dominating set problem in unit disk graphs. In: WAOA 2009, pp. 135–146 (2009)
11. Gao, X., Huang, Y., Zhang, Z., Wu, W.:  $(6 + \varepsilon)$ -approximation for minimum weight dominating set in unit disk graphs. In: Proceedings of the 14th Annual International Computing and Combinatorics Conference (COCOON 2008). LNCS, vol. 5092, pp. 551–557. Springer, Berlin (2008)
12. Garg, N., Könemann, J.: Faster and simpler algorithms for multicommodity flows and other fractional packing problems. In: Proc. 39th Annual Symposium on the Foundations of Computer Science, pp. 300–309 (1998)
13. Hochbaum, D.S., Maass, W.: Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM* **32**, 130–136 (1985)
14. Huang, Y., Gao, X., Zhang, Z., Wu, W.: A better constant-factor approximation for weighted dominating set in unit disk graph. *J. Comb. Optim.* **18**, 174–194 (2009)
15. Sorokin, A., Boyko, N., Boginski, V., Uryasev, S., Pardalos, P.: Mathematical programming techniques for sensor networks. *Algorithms* **2**, 565–581 (2009)
16. Wang, C., Willson, J., Park, M.A., Farago, A., Wu, W.: On dual power assignment optimization for biconnectivity. *J. Comb. Optim.* **19**, 174–183 (2010)
17. Willson, J.K., Ding, L., Wu, W., Wu, L., Lu, Z., Lee, W.: A better constant-approximation for coverage problem in wireless sensor networks. Preprint
18. Wu, W., Gao, X., Pardalos, P.M., Du, D.-Z.: Wireless networking, dominating and packing. *Optim. Lett.* **4**, 347–358 (2010)
19. Yang, Y., Cardei, M.: Adaptive energy efficient sensor scheduling for wireless sensor networks. *Optim. Lett.* **4**(3), 359–369 (2010)
20. Zhang, H., Hou, J.C.: Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc Sens. Wirel. Netw.* **1**, 89–124 (2005)
21. Zhang, Y., Li, W.: Modeling and energy consumption evaluation of a stochastic wireless sensor networks. Preprint (2011)
22. Zou, F., Li, X., Gao, S., Wu, W.: Node-weighted Steiner tree approximation in unit disk graphs. *J. Comb. Optim.* **18**, 342–349 (2009)
23. Zou, F., Wang, Y., Xu, X., Du, H., Li, X., Wan, P., Wu, W.: New approximations for weighted dominating sets and connected dominating sets in unit disk graphs. *Theor. Comput. Sci.* **412**(3), 198–208 (2011)