

Andrea Lodi  
Alessandro Panconesi  
Giovanni Rinaldi (Eds.)

LNCS 5035

# Integer Programming and Combinatorial Optimization

13th International Conference, IPCO 2008  
Bertinoro, Italy, May 2008  
Proceedings

 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Alfred Kobsa

*University of California, Irvine, CA, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, MA, USA*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

Andrea Lodi Alessandro Panconesi  
Giovanni Rinaldi (Eds.)

# Integer Programming and Combinatorial Optimization

13th International Conference, IPCO 2008  
Bertinoro, Italy, May 26-28, 2008  
Proceedings

## Volume Editors

Andrea Lodi  
DEIS, Università di Bologna  
40136 Bologna, Italy  
E-mail: andrea.lodi@unibo.it

Alessandro Panconesi  
Dipartimento di Informatica  
Università di Roma "La Sapienza"  
00198 Rome, Italy  
E-mail: ale@di.uniroma1.it

Giovanni Rinaldi  
Istituto di Analisi dei Sistemi  
ed Informatica "Antonio Ruberti" – CNR  
00185 Rome, Italy  
E-mail: rinaldi@iasi.cnr.it

Library of Congress Control Number: 2008927469

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

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743  
ISBN-10 3-540-68886-2 Springer Berlin Heidelberg New York  
ISBN-13 978-3-540-68886-0 Springer Berlin Heidelberg New York

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

Springer is a part of Springer Science+Business Media

[springer.com](http://springer.com)

© Springer-Verlag Berlin Heidelberg 2008  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 12278565 06/3180 5 4 3 2 1 0



# Preface

The volume contains the papers selected for presentation at IPCO 2008, the 13th International Conference on Integer Programming and Combinatorial Optimization that was held in Bertinoro (Italy), May 26–28, 2008.

The IPCO series of conferences, sponsored by the Mathematical Programming Society, highlights recent developments in theory, computation, and application of integer programming and combinatorial optimization. The first conference took place in 1990; starting from IPCO 1995, the proceedings are published in the *LNCS* series.

The 12 previous IPCO conferences were held in Waterloo (Canada) 1990, Pittsburgh (USA) 1992, Erice (Italy) 1993, Copenhagen (Denmark) 1995 [LNCS 920], Vancouver (Canada) 1996 [LNCS 1084], Houston (USA) 1998 [LNCS 1412], Graz (Austria) 1999 [LNCS 1610], Utrecht (The Netherlands) 2001 [LNCS 2081], Boston (USA) 2002 [LNCS 2337], New York (USA) 2004 [LNCS 2986], Berlin (Germany) 2005 [LNCS 3509], and Ithaca (USA) 2007 [LNCS 4168]. The conference is not held in the years when the International Symposium of the Mathematical Programming Society takes place.

A total of 95 papers were submitted, most of them of very high quality, from which 32 were selected for presentation by the Program Committee that met in Aussois (France) on January 8 and 9, 2008. The selection was based on originality and quality, and reflects many of the current directions in integer programming and combinatorial optimization research. As in the IPCO tradition, only around 30 papers could be presented at the conference, thus many submitted papers meeting high standards of originality and quality could not be selected.

On behalf of the other members of the Program Committee, we would like to thank all the external reviewers for their very valuable help in the evaluation of the papers and all the authors of the submitted papers for their support of the IPCO conferences. Finally, we like to thank the Mathematical Programming Society for the support, the conference system EasyChair that was used to handle the submissions, and the sponsors, IBM Research, ILOG, BiCi, IASI-CNR and Università di Bologna, for their generosity.

March 2008

Giovanni Rinaldi  
Andrea Lodi  
Alessandro Panconesi

# Organization

IPCO 2008 was held at the University Residential Center of Bertinoro (Forlì-Cesena), Italy, May 26–28, 2008.

## Editors

### **Andrea Lodi**

DEIS, Università di Bologna, viale Risorgimento 2, 40136 Bologna, Italy  
andrea.lodi@unibo.it

### **Alessandro Panconesi**

Dipartimento di Informatica, Università di Roma “La Sapienza”, via Salaria 113,  
00198 Rome, Italy  
ale@di.uniroma1.it

### **Giovanni Rinaldi**

Istituto di Analisi dei Sistemi ed Informatica “Antonio Ruberti” - CNR, viale  
Manzoni 30, 00185 Rome, Italy  
rinaldi@iasi.cnr.it

## Program Committee

Karen Aardal (CWI, The Netherlands)  
G rard Cornu jols (CMU, USA & LIF, Marseille, France)  
Friedrich Eisenbrand (EPFL, Lausanne, Switzerland)  
Michel X. Goemans (MIT, USA)  
Ravi Kannan (Yale University, USA)  
Andrea Lodi (Universit  di Bologna, Italy)  
Alessandro Panconesi (Universit  “La Sapienza”, Rome, Italy)  
Maurice Queyranne (University of British Columbia, Canada)  
Franz Rendl (Universit t Klagenfurt, Austria)  
Giovanni Rinaldi, Chair (IASI-CNR, Rome, Italy)  
Andr s Seb  (CNRS, Grenoble, France)  
Martin Skutella (Technische Universit t Berlin, Germany)

## Organizing Committee

Andrea Lodi (Universit  di Bologna)  
Alessandro Panconesi (Universit  “La Sapienza” di Roma)  
Andrea Tramontani (Universit  di Bologna)

## **Local Arrangements**

Andrea Bandini, Director (CeUB)

Eleonora Campori (CeUB)

Michela Schiavi (CeUB)

# Table of Contents

## Session 1

Perspective Relaxation of Mixed Integer Nonlinear Programs with Indicator Variables .....	1
Disjunctive Cuts for Non-convex Mixed Integer Quadratically Constrained Programs .....	17
The Air Traffic Flow Management Problem: An Integer Optimization Approach .....	34

## Session 2

The Induced Disjoint Paths Problem .....	47
A Weighted $K_{t,t}$ -Free $t$ -Factor Algorithm for Bipartite Graphs .....	62
A New Algorithm for the Maximum Weighted Stable Set Problem in Claw-Free Graphs .....	77
A Polynomial Algorithm for Weighted Abstract Flow .....	97

## Session 3

A Comparative Study of Linear and Semidefinite Branch-and-Cut Methods for Solving the Minimum Graph Bisection Problem .....	112
Binary Positive Semidefinite Matrices and Associated Integer Polytopes .....	125
Vertex Cover Resists SDPs Tightened by Local Hypermetric Inequalities .....	140

**Session 4**

Tight Bounds for Permutation Flow Shop Scheduling ..... 154

The Stochastic Machine Replenishment Problem ..... 169

A Polynomial Time Approximation Scheme for the Square Packing Problem ..... 184

**Session 5**

Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints ..... 199

Computing with Multi-row Gomory Cuts ..... 214

Constraint Orbital Branching ..... 225

**Session 6**

A Fast, Simpler Algorithm for the Matroid Parity Problem ..... 240

Degree Bounded Matroids and Submodular Flows ..... 259

Budgeted Matching and Budgeted Matroid Intersection Via the Gasoline Puzzle ..... 273

**Session 7**

Primal-Dual Schema for Capacitated Covering Problems ..... 288

Offline and Online Facility Leasing ..... 303

Importance Sampling via Load-Balanced Facility Location ..... 316

**Session 8**

A Constant Approximation Algorithm for the *a priori* Traveling Salesman Problem ..... 331

New Geometry-Inspired Relaxations and Algorithms for the Metric Steiner Tree Problem ..... 344

Min Sum Edge Coloring in Multigraphs Via Configuration LP ..... 359

**Session 9**

An Improved Algorithm for Finding Cycles Through Elements ..... 374

The Stable Roommates Problem with Choice Functions ..... 385

A New Approach to Splitting-Off ..... 401

**Session 10**

Can Pure Cutting Plane Algorithms Work? ..... 416

The Mixing Set with Divisible Capacities ..... 435

A Polynomial Time Algorithm for the Stochastic Uncapacitated Lot-Sizing Problem with Backlogging ..... 450

Lifting Integer Variables in Minimal Inequalities Corresponding to Lattice-Free Triangles ..... 463

**Author Index** ..... 477

# Perspective Relaxation of Mixed Integer Nonlinear Programs with Indicator Variables

Oktay Günlük<sup>1</sup> and Jeff Linderoth<sup>2</sup>

<sup>1</sup> Mathematical Sciences Department, IBM T. J. Watson Research Center,  
Yorktown Heights, NY 10598 USA

oktay@watson.ibm.com

<sup>2</sup> Department of Industrial and Systems Engineering,  
University of Wisconsin-Madison, 1513 University Avenue,

Madison, WI 53706, USA

linderoth@wisc.edu

**Abstract.** We study mixed integer nonlinear programs (MINLP) that are driven by a collection of indicator variables where each indicator variable controls a subset of the decision variables. An indicator variable, when it is “turned off”, forces some of the decision variables to assume a fixed value, and, when it is “turned on”, forces them to belong to a convex set. Most of the integer variables in known MINLP problems are of this type. We first study a mixed integer set defined by a single separable quadratic constraint and a collection of variable upper and lower bound constraints. This is an interesting set that appears as a substructure in many applications. We present the convex hull description of this set. We then extend this to produce an explicit characterization of the convex hull of the union of a point and a bounded convex set defined by analytic functions. Further, we show that for many classes of problems, the convex hull can be expressed via conic quadratic constraints, and thus relaxations can be solved via second-order cone programming. Our work is closely related with the earlier work of Ceria and Soares (1996) as well as recent work by Frangioni and Gentile (2006) and, Aktürk, Atamtürk and Gürel (2007).

Finally, we apply our results to develop tight formulations of mixed integer nonlinear programs in which the nonlinear functions are separable and convex and in which indicator variables play an important role. In particular, we present strong computational results with two applications – quadratic facility location and network design with congestion – that show the power of the reformulation technique.

## 1 Introduction

In this work, we study mixed integer nonlinear programs (MINLP) that are driven by a collection of indicator variables where each indicator variable controls a subset of the decision variables. In particular, we are interested in MINLPs where an indicator variable, when it is “turned off”, forces some of the decision variables to assume a fixed value, and, when it is “turned on”, forces them to

belong to a convex set. We call such programs  $\{ \}$ .

A generic indicator-induced  $\{0-1\}$ -MINLP can be written as

$$\min_{(x,z) \in X \times (Z \cap \mathbb{B}^p)} \{c^T x + d^T z \mid g_j(x, z) \leq 0 \ \forall j \in M, (x_{V_i}, z_i) \in S_i \ \forall i \in I\} \quad (1)$$

where  $z$  are the indicator variables,  $x$  are the continuous variables  $x_{V_i}$  denotes the collection of continuous variables (i.e.  $x_j, j \in V_i$ ), and  $C_i$  is the set of constraints controlled by the indicator variable  $z_i$ . Sets  $X \subseteq \mathbb{R}^n$  and  $Z \subseteq \mathbb{R}^p$  are polyhedral sets of appropriate dimension and  $S_i$  is the set of points that satisfy all constraints associated with the indicator variable  $z_i$ :

$$S_i \stackrel{\text{def}}{=} \left\{ (x_{V_i}, z_i) \in \mathbb{R}^{|V_i|} \times \mathbb{B} \mid \begin{array}{l} x_{V_i} = \hat{x}_{V_i} \text{ if } z_i = 0 \\ x_{V_i} \in \Gamma_i \text{ if } z_i = 1 \end{array} \right\},$$

where

$$\Gamma_i \stackrel{\text{def}}{=} \{x_{V_i} \in \mathbb{R}^{|V_i|} \mid f_j(x_{V_i}) \leq 0 \ \forall j \in C_i, u_k \geq x_k \geq \ell_k \ \forall k \in V_i\}.$$

is bounded for all  $i \in I$ . The objective function in (1) is assumed to be linear without loss of generality (if necessary, an additional variable can be used to move the nonlinearity from the objective function to the constraint set.)

In this paper we study the convex hull description of the sets  $S_i$  when  $\Gamma_i$  is a convex set. Note that  $\Gamma_i$  can be convex even when some of the  $f_j$  defining it are non-convex. Let  $S_i^c = \text{conv}(S_i)$ . Using  $S_i^c$ , one can write a “tight” continuous relaxation of (1)

$$\min_{(x,z) \in X \times Z} \{c^T x + d^T z \mid g_j(x, z) \leq 0 \ \forall j \in M, (x_{V_i}, z_i) \in S_i^c \ \forall i \in I\} \quad (2)$$

where  $S_i$  in (1) is replaced by its convex hull and integrality requirement on  $z$  is dropped. We assume that  $Z$  already contains bound constraints for  $z$ . We call (2), the relaxation of (1) as description of  $S_i^c$  involves functions which we discuss later. We also present computational results and show that (2) indeed gives a strong relaxation when applied to a number of problems. We also show that in some cases,  $S_i^c$  is representable as a second-order cone and this improves computational effectiveness of our approach even further.

Indicator-induced MINLPs can be used to model many interesting problems. Two applications that we study in detail in this paper are: (i) the quadratic-cost uncapacitated facility location problem recently introduced in [1], and, (ii) network design problem under queuing delay, first discussed in [2]. For other examples see [3,4,5] for portfolio optimization problems; or, [6] for a job-scheduling problem with controllable processing times. In addition, certain classes of unit commitment problems for electrical power generation can be formulated as indicator-induced MINLPs.



There has been some recent work on generating strong relaxations for convex MINLPs. One line of work has been on extending general classes of cutting planes from mixed integer linear programs. Specifically, [7] explain how the disjunctive cutting planes of [8] can be applied for MINLP, [9] extend the Gomory cutting plane procedure [10], and [11] extend the mixed integer rounding procedure of [12] to conic mixed integer programs. A second line of work has focused on generating problem specific cutting planes, for example see [1] for different families of inequalities for a quadratic cost facility location problem. In some cases these inequalities can be used to strengthen the perspective relaxation even further.

There are two recent papers that are closely related with our work. Frangioni and Gentile [13] have introduced a class of linear inequalities called  $\bullet$  for indicator-induced MINLPs. As we discuss later, perspective cuts are essentially outer approximation cuts for  $S_i^c$  and therefore the perspective relaxation (2) can be viewed as implicitly including all (infinitely many) perspective cuts to a straightforward relaxation of (1). Very recently, Aktürk, Atamtürk, and Gürel [6] independently gave a strong characterization of  $S_i^c$  when  $\Gamma_i = \{x \in \mathbb{R}^2 \mid x_1^t - x_2 \leq 0, u \geq x_1, x_2 \geq 0\}$  for  $t \geq 1$ . They use this characterization in an algorithm for solving some classes of nonlinear machine scheduling problems.

## 2 A Quadratic Set with Variable Bounds

In this section we present a convex hull description of the following set

$$Q = \left\{ (w, x, z) \in \mathbb{R} \times \mathbb{R}^n \times \mathbb{B}^n : w \geq \sum_{i \in N} q_i x_i^2, u_i z_i \geq x_i \geq l_i z_i, x_i \geq 0, \forall i \in N \right\},$$

where  $N = \{1, 2, \dots, n\}$ ,  $q_i \in \mathbb{R}_+$  and  $u_i, l_i \in \mathbb{R}$  for all  $i = 1, 2, \dots, n$ . We then use this insight to define the convex hull of more complicated mixed integer nonlinear sets. Set  $Q$  appears in a number of non-linear mixed-integer programs as a substructure and we present some examples of this in Section 4. To our knowledge, the first convex hull description of  $Q$  was stated without proof in the unpublished Ph.D. thesis [14].

### 2.1 A Simple Set

To understand the set  $Q$ , we first study a simpler mixed-integer set with only 3 variables, which can be obtained by setting  $n = 1$  and  $q_1 = 1$ .

Let

$$S = \left\{ (x, y, z) \in \mathbb{R}^2 \times \mathbb{B} : y \geq x^2, uz \geq x \geq lz, x \geq 0 \right\}$$

where  $u, l \in \mathbb{R}$ . We next show that the convex hull of  $S$  is given by

$$S^c = \left\{ (x, y, z) \in \mathbb{R}^3 : yz \geq x^2, uz \geq x \geq lz, 1 \geq z \geq 0, x, y \geq 0 \right\}.$$

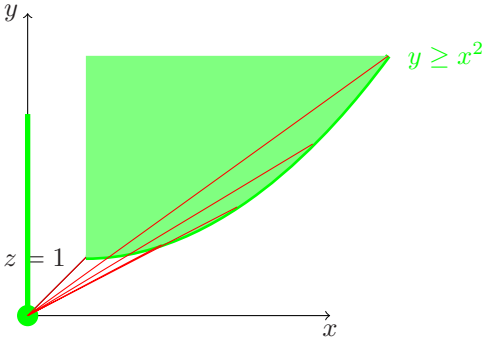


Fig. 1. The set  $S^c$

Note that even though  $yz \geq x^2$  is not a convex constraint (as its Hessian is not positive semi-definite), it still defines a convex set in  $\mathbb{R}_+^3$ .

**Lemma 1.**  $\text{conv}(S) = S^c$

Geometrically, the set  $S^c$  consists of all points that lie above a line segment connecting the origin to the point  $(t, t^2, 1)$  for each  $t \geq 0$ , as shown in Figure 1.

### 2.2 An Extended Formulation

Consider the following extended formulation of  $Q$

$$\bar{Q} = \left\{ (w, x, y, z) \in \mathbb{R}^{1+n+n+n} : w \geq \sum_i q_i y_i, (x_i, y_i, z_i) \in S_i, \forall i \in N \right\},$$

where  $S_i$  has the same form as the set  $S$  discussed in the previous section except the bounds  $u$  and  $l$  are replaced with  $u_i$  and  $l_i$ . Note that if  $(w, x, y, z) \in \bar{Q}$  then  $(w, x, z) \in Q$ , and therefore  $\text{proj}_{(w,x,z)}(\bar{Q}) \subseteq Q$ . On the other hand, for any  $(w, x, z) \in Q$ , letting  $y'_i = x_i^2$  gives a point  $(w, x, y', z) \in \bar{Q}$ . Therefore,  $\bar{Q}$  is indeed an extended formulation of  $Q$ , or, in other words,  $Q = \text{proj}_{(w,x,z)}(\bar{Q})$ .

Before we present a convex hull description of  $\bar{Q}$  we first define some basic properties of mixed-integer sets which are not necessarily polyhedral. Using these definitions, we then show some elementary observations which are known for polyhedral sets.

**Definition 1.**  $P \subset \mathbb{R}^n$  extreme point  $p \in P$   $p = 1/2p_1 + 1/2p_2$   $p_1, p_2 \in P$   $p_1 \neq p_2$  pointed

**Definition 2.**  $P \subset \mathbb{R}^n$  integral  $I \subseteq \{1, \dots, n\}$   $p \in P$   $p_i \in \mathbb{Z}$   $i \in I$

**Lemma 2.**  $i = 1, 2$   $P_i \subset \mathbb{R}^{n_i}$   $P' = \{(x, y) \in \mathbb{R}^{n_1+n_2} : x \in P_1, y \in P_2\}$

$\text{conv}(P') = \{(x, y) \in \mathbb{R}^{n_1+n_2} : x \in \text{conv}(P_1), y \in \text{conv}(P_2)\}$ . ■

**Lemma 3.**  $\dots P \subset \mathbb{R}^n \dots P' = \{(w, x) \in \mathbb{R}^{n+1} : w \geq ax, x \in P\} \dots a \in \mathbb{R}^n$

$$\dots \text{conv}(P') = P'' \dots P'' = \{(w, x) \in \mathbb{R}^{n+1} : w \geq ax, x \in \text{conv}(P)\}. \quad \blacksquare$$

We are now ready to present the convex hull of  $\bar{Q}$ . To that end, consider the set

$$\bar{Q}^c = \left\{ (w, x, y, z) \in \mathbb{R}^{1+n+n+n} : w \geq \sum_i q_i y_i, (x_i, y_i, z_i) \in S_i^c, \forall i \in N \right\}.$$

**Lemma 4.**  $\dots \bar{Q}^c \dots \text{conv}(\bar{Q}) = \bar{Q}^c$

Let  $D = \{x \in \mathbb{R}^n, y \in \mathbb{R}^n, z \in \times \mathbb{R}^n : (x_i, y_i, z_i) \in S_i, i = 1, 2, \dots, n\}$  so that  $\bar{Q} = \{w \in \mathbb{R}, x \in \mathbb{R}^n, y \in \mathbb{R}^n, z \in \times \mathbb{R}^n : w \geq \sum_{i=1}^n q_i y_i, (x, y, z) \in D\}$ . By Lemma 3, the convex hull of  $\bar{Q}$  can be obtained by replacing  $D$  with its convex hull in this description. By Lemma 2, this can simply be done by taking convex hulls of  $S_i$ 's, that is, by replacing  $S_i$  with  $\text{conv}(S_i)$  in the description of  $D$ . Finally, by Lemma 3,  $\bar{Q}^c$  is integral.

### 2.3 Convex Hull Description in the Original Space

Let

$$Q^c = \left\{ (w, x, z) \in \mathbb{R}^{1+n+n} : w \prod_{i \in S} z_i \geq \sum_{i \in S} (q_i x_i^2 \prod_{l \in S \setminus \{i\}} z_l), S \subseteq N \right. \\ \left. u_i z_i \geq x_i \geq l_i z_i, x_i \geq 0, \forall i \in N \right\}. \quad (II)$$

Notice that a given point  $\bar{p} = (\bar{w}, \bar{x}, \bar{z})$  satisfies inequality (II) for a particular  $S \subseteq N$  if and only if one of the following conditions hold: (i)  $\bar{z}_i = 0$  for some  $i \in S$ , or, (ii) if all  $z_i > 0$ , then  $\bar{w} \geq \sum_{i \in S} q_i \bar{x}_i^2 / \bar{z}_i$ . Based on this observation we next show that these (exponentially many) inequalities are sufficient to describe the convex hull of  $Q$  in the space of the original variables.

**Lemma 5.**  $Q^c = \text{proj}_{(w,x,z)}(\bar{Q}^c)$ .

Let  $\bar{p} = (\bar{w}, \bar{x}, \bar{y}, \bar{z}) \in \bar{Q}^c$  and define  $S(\bar{p}) = \{i : z_i > 0\}$ . Clearly  $u_i \bar{z}_i \geq \bar{x}_i \geq l_i \bar{z}_i$  and  $\bar{x}_i \geq 0$  for all  $i = 1, 2, \dots, n$ . Furthermore, inequality (II) is satisfied for all  $S$  such that  $S \not\subseteq S(\bar{p})$ . In addition, notice that, as  $q \geq 0$ ,

$$\bar{w} \geq \sum_{i \in S(\bar{p})} q_i \bar{y}_i \geq \sum_{i \in S(\bar{p})} q_i \bar{x}_i^2 / \bar{z}_i \geq \sum_{i \in S'} q_i \bar{x}_i^2 / \bar{z}_i$$

for all  $S' \subseteq S(\bar{p})$ . Therefore  $\bar{p}$  satisfies inequality (II) for all  $S$  and  $\text{proj}_{(w,x,z)}(\bar{Q}^c) \subseteq Q^c$ .

Next, let  $\bar{p} = (\bar{w}, \bar{x}, \bar{z}) \in Q^c$  be given and let

$$\bar{y}_i = \begin{cases} 0 & \bar{z}_i = 0 \\ \bar{x}_i^2 / \bar{z}_i & \text{otherwise.} \end{cases}$$

It is easy to see that  $(\bar{x}_i, \bar{y}_i, \bar{z}_i) \in S_i$  for all  $i \in \{1, 2, \dots, n\}$ . Furthermore,

$$\bar{w} \geq \sum_{i \in S(\bar{p})} q_i \bar{x}_i^2 / \bar{z}_i = \sum_{i \in S(\bar{p})} q_i \bar{y}_i = \sum_{i=1}^n q_i \bar{y}_i$$

implying that  $(\bar{w}, \bar{x}, \bar{y}, \bar{z}) \in \bar{Q}^c$  and therefore  $Q^c \subseteq \text{proj}_{(w,x,z)}(\bar{Q}^c)$ .

## 2.4 SOCP Representation

A second-order cone constraint is a constraint of the form

$$\|Ax + b\|_2 \leq c^T x + d. \quad (3)$$

The set of points  $x$  that satisfy (3) forms a convex set, and efficient and robust algorithms exist for solving optimization problems containing second-order cone constraints [15,16]. An interesting and important observation from a computational standpoint is that the nonlinear inequalities in the definitions of the sets  $S^c$  and  $\bar{Q}^c$  can be written as second-order cone constraints. All the nonlinear constraints in the definition  $S^c$  and  $\bar{Q}^c$  are of the simple form

$$x^2 \leq yz \text{ with } y \geq 0, z \geq 0, \quad (4)$$

and this is algebraically equivalent to the second-order cone constraint

$$\|(2x, y - z)^T\| \leq y + z.$$

Constraints of the form (4) are often called *rotated second-order cone* constraints. The computational benefit of dealing with inequalities (4) as second-order cone constraints rather than general nonlinear constraints will be demonstrated in Section 4.1.

## 3 A Generalization and Connections to Previous Work

We next extend the observations presented in Section 2 to describe the convex hull of a point  $\bar{x} \in \mathbb{R}^n$  and a bounded convex set defined by analytic functions. In other words, using an indicator variable  $z \in \{0, 1\}$ , define

$$W^0 = \{(x, z) \in \mathbb{R}^{n+1} : x = \bar{x}, z = 0\} \text{ and} \\ W^1 = \{(x, z) \in \mathbb{R}^{n+1} : f_i(x) \leq 0 \text{ for } i \in I, u \geq x - \bar{x} \geq l, z = 1\},$$

where  $u, l \in \mathbb{R}_+^n$ , and  $I = \{1, \dots, t\}$ . We are interested in the convex hull of  $W = W^1 \cup W^0$ . Clearly, both  $W^0$  and  $W^1$  are bounded and  $W^0$  is a convex set. Furthermore, if  $W^1$  is also convex then

$$\text{conv}(W) = \{p \in \mathbb{R}^{n+1} : p = \alpha p^1 + (1 - \alpha)p^0, p^1 \in W^1, p^0 \in W^0, 1 \geq \alpha \geq 0\}.$$

We next present a description of  $\text{conv}(W)$  in the space of original variables.

### 3.1 Reformulation in the Original Space

To simplify notation we assume that  $\bar{x} = 0$  in the remainder of this section. Note that there is no loss of generality as this is an affine transformation. We next write the description of  $\text{conv}(W)$  in open form

$$\begin{aligned} \text{conv}(W) = \left\{ (x, z) \in \mathbb{R}^{n+1} : \right. & \quad 1 \geq \alpha \geq 0, \\ & \quad x = \alpha x^1 + (1 - \alpha)x^0, \quad z = \alpha z^1 + (1 - \alpha)z^0, \\ & \quad x^0 = \bar{x}, \quad z^0 = 0, \\ & \left. f_i(x^1) \leq 0 \text{ for } i \in I, \quad u \geq x^1 - \bar{x} \geq l, \quad z^1 = 1 \right\}. \end{aligned} \tag{XF}$$

The additional variables used in this description can be projected out to obtain a description in the space of the original variables.

**Lemma 6.**  $W^1 \cup W^0 = \text{conv}(W) = W^- \cup W^0$

$$W^- = \left\{ (x, z) \in \mathbb{R}^{n+1} : f_i(x/z) \leq 0, \quad i \in I, \quad uz \geq x \geq lz, \quad 1 \geq z > 0 \right\}.$$

As  $x^0, z^0$  and  $z^1$  are fixed in (XF), it is possible to substitute out these variables. In addition, as  $z = \alpha$  after these substitutions, we can eliminate  $\alpha$ . Furthermore, as  $x = \alpha x^1 = z x^1$ , we can eliminate  $x^1$  by replacing it with  $x/z$  provided that  $z > 0$ . If, on the other hand,  $z = 0$ , clearly  $(x, 0) \in \text{conv}(W)$  if and only if  $(x, 0) \in W^0$ .

We next show that  $W^0$  is contained in the closure of  $W^-$ .

**Lemma 7.**  $1 \geq z > 0, Q^c(z) = \{x \in \mathbb{R}^n : f_i(x/z) \leq 0, \quad i \in I, \quad uz \geq x \geq lz\}$

$$\lim_{z \rightarrow 0^+} Q^c(z) = \{x \in \mathbb{R}^n : x = 0\}$$

Let  $\{z_k\} \subset (0, 1)$  be a sequence converging to 0. As, by definition,  $Q^c(z) \neq \emptyset$  for  $z \in (0, 1)$ , there exists a corresponding sequence  $\{x_k\}$  such that  $x_k \in Q^c(z_k)$ . Clearly,  $uz \geq x_k \geq lz$  and therefore  $\{x_k\}$  converges to 0.

Combining the previous lemmas, we obtain the following result.

**Corollary 1.**  $\text{conv}(W) = \text{closure}(W^-)$

We would like to emphasize that even when  $f(x)$  is a convex function  $f_i(x/z)$  may not be convex. However, for  $z > 0$  we have

$$f_i(x/z) \leq 0 \iff z^t f_i(x/z) \leq 0 \tag{5}$$

for any  $t \in \mathbb{R}$ . In particular, taking  $t = 1$  gives  $z f_i(x/z)$  which is known to be convex provided that  $f(x)$  is convex. We discuss this further in Section 3.2.

We also note that if  $f(x)$  is SOCP-representable, then  $zf_i(x/z)$  is also SOCP-representable [17]. In particular, if  $W^1$  is defined by SOCP-representable functions, then so is  $\text{conv}(W)$ . We will show the benefits of employing SOC solvers for (non-quadratic) SOC-representable sets in Section 4.2.

We next show that when all  $f_i(x)$  that define  $W^1$  are polynomial functions, convex hull of  $W$  can be described explicitly.

**Lemma 8.**  $f_i(x) = \sum_{t=1}^{p_i} c_{it} \prod_{j=1}^n x_j^{q_{itj}}$ ,  $i \in I$ ,  $q_{it} = \sum_{j=1}^n q_{itj}$ ,  $q_i = \max_t \{q_{it}\}$ ,  $f_i(x) \in [l, u]$ ,  $\text{conv}(W) = W^c$

$$W^c = \left\{ (x, z) \in \mathbb{R}^{n+t+1} : \sum_{t=1}^{p_i} c_{it} z^{q_i - q_{it}} \prod_{j=1}^n x_j^{q_{itj}} \leq 0, \quad i \in I, \right. \\ \left. zu \geq x \geq lz, \quad 1 \geq z \geq 0, \right\}.$$

Note that  $f_i(x/z) = \sum_{t=1}^{p_i} c_{it} z^{-q_{it}} \prod_{j=1}^n x_j^{q_{itj}}$ . Therefore, multiplying  $f_i(x/z) \leq 0$  by  $z^{q_i}$ , one obtains the expression above. Clearly,  $W^c \cap \{z > 0\} = W^-$  and  $W^c \cap \{z = 0\} = W^0$ .

### 3.2 Convex Hulls of Convex Sets

Given a collection of bounded convex sets, it is easy to define an extended formulation to describe their convex hull using additional variables, similar to (XF). It is however, not possible to produce a description in the space of original variables. The particular case we considered in the previous section involves only two sets, one of which consists of a single point. For the sake of completeness we next summarize some related results from [18].

Ceria and Soares [18] use perspective functions of the functions that define the original sets to produce an extended formulation for the convex hull description. If the original sets are defined by convex functions, their perspective functions are also convex. More precisely, for  $t = 1, \dots, p$ , let  $G^t : \mathbb{R}^n \rightarrow \mathbb{R}^{m_t}$  be a mapping defined by convex functions and assume that the corresponding set

$$K^t = \{x \in \mathbb{R}^n : G^t(x) \leq 0\}$$

is bounded. Let  $\tilde{G}^t : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^{m_t}$  be the perspective mapping defined as

$$\tilde{G}^t(\lambda, x) = \begin{cases} \lambda G^t(x/\lambda) & \text{if } \lambda > 0 \\ 0 & \text{if } \lambda = 0 \\ \infty & \text{otherwise} \end{cases}$$

We next state an important observation from [18] that shows the use of perspective functions to obtain convex hulls of convex sets.

**Lemma 9** ([18]).  $K^t$ ,  $t \in T = \{1, \dots, T\}$ ,  $K = \text{conv}(\cup_{t=1}^T K^t)$ ,  $x \in K$

$$x = \sum_{t=1}^{|T|} x^t ; \quad \tilde{G}^t(\lambda_t, x^t) \leq 0, \quad \sum_{t=1}^{|T|} \lambda_t = 1, \quad \lambda_t \geq 0, \quad t \in T$$

Put into this context, our observations in Section 3.1 specialize Lemma 9 to the case when  $|T| = 2$  and one of the sets contain a single point. In this special case Corollary 1 and Lemma 8 show that a description of the convex hull in the original space can be obtained easily.

### 3.3 Perspective Cuts

Building on the work [18], Frangioni and Gentile [13] introduce the class of  $\bullet$  for mixed integer programs of the form

$$\min_{(x,z) \in \mathbb{R}^n \times \mathbb{B}} \left\{ f(x) + cz \mid Ax \leq bz \right\},$$

where (i)  $X = \{x \mid Ax \leq b\}$  is bounded (also implying  $\{x \mid Ax \leq 0\} = \{0\}$ ), (ii)  $f(x)$  is a convex function that is finite on  $X$ , and (iii)  $f(0) = 0$ . Under these assumptions, they are able to show that for any  $\bar{x} \in X$  and  $s \in \partial f(\bar{x})$ , the following (linear) inequality

$$v \geq f(\bar{x}) + c + s^T(x - \bar{x}) + (c + f(\bar{x}) - s^T\bar{x})(z - 1) \tag{6}$$

is valid for the equivalent mixed integer program

$$\min_{(x,z,v) \in \mathbb{R}^n \times \mathbb{B} \times \mathbb{R}} \left\{ v \mid v \geq f(x) + cz, Ax \leq bz \right\}.$$

The inequalities (6) are derived from a first-order analysis of the convex envelope of the perspective function of  $f(x)$ .

A similar first-order argument can be used to derive inequality (6) from the characterization of the convex hull of the union of a convex set and a point given in Section 3. First define  $P^0 \stackrel{\text{def}}{=} \{(x, z, v) \in \mathbb{R}^{n+2} : x = 0, z = 0, v = 0\}$ , and

$$P^1 \stackrel{\text{def}}{=} \{(x, z, v) \in \mathbb{R}^{n+2} : Ax \leq b, f(x) + c - v \leq 0, u_x \geq x \geq l_x, u_v \geq v \geq l_v, z = 1\}$$

where bounds on variables  $x$  and  $v$  are introduced without loss of generality. Corollary 1 states that  $\text{conv}(P^0 \cup P^1)$  is the closure of

$$P^- \stackrel{\text{def}}{=} \left\{ (x, z, v) \in \mathbb{R}^{n+2} \mid Ax \leq b, zf(x/z) + cz - v \leq 0, \right. \\ \left. u_x z \geq x \geq l_x z, u_v z \geq v \geq l_v z, 1 \geq z \geq 0 \right\}.$$

For any  $\bar{z} > 0$ , a first-order (outer)-approximation of the nonlinear constraint  $zf(x/z) + cz - v \leq 0$  about the point  $(\bar{x}, \bar{z}, \bar{v})$  gives

$$0 \geq \bar{z}f(\bar{x}/\bar{z}) + c\bar{z} - \bar{v} + \begin{bmatrix} s \\ (-1/\bar{z})\bar{x}^T s_{x/z} + f(\bar{x}/\bar{z}) + c \\ -1 \end{bmatrix}^T \begin{bmatrix} x - \bar{x} \\ z - \bar{z} \\ v - \bar{v} \end{bmatrix},$$

where  $s \in \partial f(\bar{x})$  and  $s_{x/z} \in \partial f(\bar{x}/\bar{z})$ . Taking  $\bar{z} = 1$ ,  $\bar{v} = f(\bar{x}) + c$ , and rearranging terms gives inequality (6) above.

## 4 Applications

In this section, two applications are described: a quadratic uncapacitated facility location problem and a network design problem with nonlinear congestion constraints. In each case, the positive impact of the perspective reformulation and the ability to model the nonlinear inequalities in the reformulations as second-order cone constraints is demonstrated.

### 4.1 Separable Quadratic UFL

The Separable Quadratic Uncapacitated Facility Location Problem (SQUFL) was introduced by [11]. In the SQUFL, there is a set of customers ( $N = \{1, 2, \dots, n\}$ ), a set of facilities ( $M = \{1, 2, \dots, m\}$ ), and each customer must have its demand for a single commodity met by an open facility. There is a fixed cost  $c_i$  for opening a facility  $i \in M$ . Meeting the demand of customer  $j \in N$  from facility  $i \in M$  costs an amount proportional to the square of the quantity delivered. A mixed integer nonlinear program for the SQUFL is

$$z^* \stackrel{\text{def}}{=} \min_{(x,z) \in \mathbb{R}_+^{mn} \times \mathbb{B}^m} \left\{ \sum_{i \in M} c_i z_i + \sum_{i \in M} \sum_{j \in N} q_{ij} x_{ij}^2 \mid x_{ij} \leq z_i \ \forall i \in M, \forall j \in N, \right. \\ \left. \sum_{i \in M} x_{ij} = 1 \ \forall j \in N \right\}. \quad (7)$$

The variables  $z_i$  indicate if facility  $i \in N$  is open, and  $x_{ij}$  is a decision variable representing the fraction of customer  $j$ 's demand met from facility  $i$ . We let  $z_R$  be the optimal solution value of the relaxation of (7) in which the constraints  $z_i \in \{0, 1\}$  are replaced by  $z_i \in [0, 1]$ .

To write SQUFL as an indicator-induced MINLP, the auxiliary variables  $y_{ij} \ \forall i \in M, j \in N$  are introduced. The objective function is changed to the linear function

$$\min \sum_{i \in M} c_i z_i + \sum_{i \in M} \sum_{j \in N} q_{ij} y_{ij},$$

and the constraints

$$x_{ij}^2 - y_{ij} \leq 0 \quad \forall i \in M, j \in N \quad (8)$$

are added. In this reformulation, if the indicator variable  $z_i = 0$ , then  $x_{ij} = 0 \ \forall j \in N$  and the constraints (8) become redundant, while if  $z_i = 1$ , the constraints (8) become active. Thus, the constraints (8) can be replaced by their perspective counterparts

$$x_{ij}^2 - z_i y_{ij} \leq 0 \quad \forall i \in M, \forall j \in N, \quad (9)$$

and the resulting relaxation should be significantly tighter. We will let  $z_P$  denote the optimal solution value of the relaxation of the perspective reformulation.



**Computational Results.** To test the strength of the perspective reformulation, random instances were constructed with facilities and locations uniformly distributed in the unit square. The fixed cost of opening facility  $i \in M$  was taken to be  $c_i = \lfloor \mathcal{U}(1, 100) \rfloor$ . If  $p_i \in [0, 1]^2$  was the location of facility  $i \in M$  and  $r_j \in [0, 1]^2$  was the location of customer  $j \in N$ , then the variable cost parameter was calculated as  $q_{ij} = 50\|p_i - r_j\|$ .

Instances are constructed in a similar manner in [1]. For  $m \in \{10, 20, 30, 20\}$  and  $n \in \{30, 50, 100, 200\}$ , ten instances were created and solved using the non-linear branch-and-bound algorithm available in the open-source MINLP code BONMIN [19]. The instances were solved using both the original formulation (7) and the perspective reformulation. All instances were solved on a 1.8GHz AMD Opteron CPU.

Table 1 shows the results of this experiment. In the table,  $\bar{z}_R$  represents the average value of the relaxation of the original formulation,  $\bar{z}_P$  the average value of the relaxation of the perspective reformulation, and  $\bar{z}^*$  the average value of the optimal solution found by BONMIN. The table also displays the number of instances (out of 10) that were solved within a time limit of 8 hours, the average number of nodes  $\bar{N}$  required to solve the instances, and the average CPU time ( $\bar{T}$ ) in seconds for both the original and perspective formulations. Clearly, reformulating the problem via the perspective reformulation has an enormous impact on the ability to solve the problem.

The results in Table 1 indicate that the CPU time required to solve one node of the branch-and-bound tree increases dramatically when the perspective formulation is applied. BONMIN uses the interior-point solver Ipopt [20] for solving relaxations that arise at nodes of the branch-and-bound tree. Ipopt is a solver

**Table 1.** Relaxation Values and Solution Times for SQUFL

$m$	$n$				Original Formulation			Perspective Formulation		
		$\bar{z}_R$	$\bar{z}_P$	$\bar{z}^*$	# Solved	$\bar{N}$	$\bar{T}$	# Solved	$\bar{N}$	$\bar{T}$
10	30	105.8	196.5	197.9	10	333	8.9	10	15	3.7
10	50	160.4	312.6	314.6	10	406	18.0	10	11	4.9
10	100	266.5	460.4	462.0	10	441	36.7	10	9	7.7
10	200	470.7	733.6	737.0	10	350	59.7	10	7	15.2
20	30	81.7	186.1	185.6	10	3452	213.7	10	37	39.9
20	50	111.6	274.8	276.2	10	5526	601.4	10	31	85.9
20	100	166.3	412.7	414.5	7	25901	12263.9	10	35	677.1
20	200	283.5	650.8	653.1	0	-	-	10	27	1925
30	30	64.1	157.8	159.4	9	17837	1822.7	10	62	192.8
30	50	82.1	241.6	243.3	1	61062	23760.2	10	56	650.3
30	100	126.0	343.4	345.6	0	-	-	10	51	4565.4
30	200	200.7	545.8	547.4	0	-	-	9	44	16858.5
40	30	58.6	146.4	147.7	7	55660	9319.6	10	71	224.3
40	50	74.1	198.7	200.0	0	-	-	10	85	3030.6
40	100	109.6	309.8	311.2	0	-	-	10	64	8420.8
40	200	161.4	478.3	-	0	-	-	0	-	-

for general nonlinear programs and is unable to exploit the special second-order cone structure of the inequalities in the perspective reformulation. Even more disturbing is the fact that since the functions  $x^2 - yz$  appearing in the perspective reformulation are not convex, Ipopt cannot guarantee convergence to a stationary point and its performance is highly dependent on the quality of the initial iterate provided.

To eliminate the obstacles faced by a general NLP solver, the conic formulations were solved with Mosek [16], a code specialized for problems of this type. Table 2 shows the number of nodes ( $N$ ) and CPU seconds ( $T$ ) required by Mosek v5.0 to solve large random instances of SQUFL formulated with the perspective reformulation wherein the nonlinear inequalities are represented in second-order-cone form. Note the order-of-magnitude improvement in solution time, which comes solely from the reduced time to solve relaxations at nodes of the branch-and-bound tree.

Table 3, taken from Table 1 of the paper of [1], shows the effectiveness of three classes of cutting planes introduced there at closing the optimality gap at the root node. In the table  $z_R$  is the value of the relaxation of the original formulation,  $z_{GLW}$  is the value of the relaxation with three classes of valid inequalities added,  $z_P$  is the value of the relaxation of the perspective reformulation, and  $z^*$  is the optimal solution value. The table shows that the perspective reformulation is significantly better at closing the integrality gap than are the cutting planes of [1].

**Table 2.** SOCP Solution Times

$m$	$n$	$T$	$N$
30	200	141.9	63
40	100	76.4	54
40	200	101.3	45
50	100	61.6	49
50	200	140.4	47

**Table 3.** Relaxation Bounds for SQUFL

$m$	$n$	$z_R$	$z_{GLW}$	$z_P$	$z^*$
10	30	140.6	326.4	346.5	348.7
15	50	141.3	312.2	380.0	384.1
20	65	122.5	248.7	288.9	289.3
25	80	121.3	260.1	314.8	315.8
30	100	128.0	327.0	391.7	393.2

The largest of the instances in Table 3 was solved to optimality by Lee [21] using BONMIN. The solution required 16697 CPU seconds and 45,901 nodes for the original formulation, and a 21206 CPU seconds and 29277 nodes for the formulation with additional inequalities added. The same instance was solved using Mosek v5 on the perspective reformulation wherein the nonlinear inequalities were written as second-order cone constraints. Solution of the instance required only 44 branch-and-bound nodes and  $\dots$  to solve on Intel Pentium 4 CPU with a clock speed of 2.60GHz, a speedup factor of more than 700.

## 4.2 Network Design with Congestion Constraints

In this section, a model for constructing a communication network at minimum cost meeting a design specification for total queuing delay is presented. Similar

models appear in the works [2,22,23]. In the problem, there is a set of commodities  $K$  to be shipped over a capacitated directed network  $G = (N, A)$ . The capacity of arc  $(i, j) \in A$  is  $u_{ij}$ , and each node  $i \in N$  supplies or demands a specified amount  $b_i^k$  of commodity  $k$ . There is a fixed cost  $c_{ij}$  of opening each arc  $(i, j) \in A$ , and we introduce  $\{0-1\}$  decision variables  $z_{ij}$  to indicate whether arc  $(i, j) \in A$  is opened. The quantity of commodity  $k$  routed on arc  $(i, j)$  is measured by the decision variable  $x_{ij}^k$ . A typical function to measure the total weighted congestion (or queuing delay) of a flow  $f_{ij} = \sum_{k \in K} x_{ij}^k$  in the network is

$$\rho(f) \stackrel{\text{def}}{=} \sum_{(i,j) \in A} r_{ij} \frac{f_{ij}}{1 - f_{ij}/u_{ij}},$$

where  $r_{ij} \geq 0$  is a user-defined importance parameter for the queuing delay that occurs on arc  $(i, j)$ . We use a decision variables  $y_{ij}$  to measure the contribution of the congestion on arc  $(i, j)$  to the total congestion  $\rho(f)$ . The network should be designed so as to keep the total queuing delay less than a given value  $\beta$ , and this is to be accomplished at minimum cost. The resulting optimization model can be written as

$$\begin{aligned} & \min_{(x,y,z,f) \in \mathbb{R}_+^{|A| \times |K|} \times \mathbb{R}_+^{|A|} \times \mathbb{B}^{|A|} \times \mathbb{R}_+^{|A|}} \sum_{(i,j) \in A} c_{ij} z_{ij} \\ & \text{subject to} \quad \sum_{(j,i) \in A} x_{ij}^k - \sum_{(i,j) \in A} x_{ij}^k = b_i^k \quad \forall i \in N, \forall k \in K \\ & \quad \quad \quad \sum_{k \in K} x_{ij}^k - f_{ij} = 0 \quad \forall (i, j) \in A \\ & \quad \quad \quad f_{ij} \leq u_{ij} z_{ij} \quad \forall (i, j) \in A \quad (10) \\ & \quad \quad \quad y_{ij} \geq \frac{r_{ij} f_{ij}}{1 - f_{ij}/u_{ij}} \quad \forall (i, j) \in A \quad (11) \\ & \quad \quad \quad \sum_{(i,j) \in A} y_{ij} \leq \beta \end{aligned}$$

An observation not previously made in the literature regarding this network design problem is that the congestion inequalities (10) can be written as second-order cone constraints. Multiplying both sides of the inequality by  $1 - f_{ij}/u_{ij} > 0$ , adding  $r_{ij} f_{ij}^2$  to both sides of the inequality, and factoring the left-hand-side gives an equivalent constraint

$$(y_{ij} - r_{ij} f_{ij})(u_{ij} - f_{ij}) \geq r_{ij} f_{ij}^2. \quad (12)$$

Because  $y_{ij} \geq r_{ij} f_{ij}$  and  $u_{ij} \geq f_{ij}$ , (12) is precisely a constraint in rotated second-order conic form (4).

The relaxation can be strengthened by noting that if  $z_{ij} = 0$ , then the constraints (10) force  $f_{ij} = 0$ , and the constraints (11) are redundant for the arc  $(i, j)$ . However, if  $z_{ij} = 1$ , then the definitional constraint (10) for the corresponding  $y_{ij}$  must hold. We can then strengthen the formulation by applying

the perspective reformulation. Specifically, each constraint (11) can be replaced by its perspective counterpart:

$$z_{ij} \left[ \frac{r_{ij} f_{ij} / z_{ij}}{1 - f / (u_{ij} z_{ij})} - \frac{y_{ij}}{z_{ij}} \right] \leq 0. \quad (13)$$

The constraints (13) can also be written as second order cone constraints in a similar fashion to the non-perspective version (11). Specifically, simplifying the left-hand side of the inequality (13), adding  $r_{ij} f_{ij}^2$  to both sides of the simplified inequality and factoring gives the equivalent constraints

$$(y_{ij} - r_{ij} f_{ij})(u_{ij} z_{ij} - f_{ij}) \geq r_{ij} f_{ij}^2,$$

which is a rotated second-order cone constraint since  $y_{ij} \geq r_{ij} f_{ij}$  and  $u_{ij} z_{ij} \geq f_{ij}$ . The fact that the inequalities in the perspective reformulation of (11) are SOC-representable is no surprise. In fact, [17] (Page 96, Proposition 3.3.2) show that the perspective transformation of a function whose epigraph is a SOC-representable set is nearly always SOC-representable.

**Computational Results.** To assess the strength of the perspective reformulation of this nonlinear network design problem, three test instances were created. The first instance was the **atlanta** network from SNDLIB [24]. The second and third instances were generated randomly. MPS files for all of the instances are available on request from the authors.

Each of the instances in the test suite was solved using Mosek v5.0 using both the original and perspective formulations of the problem on an Intel Pentium 4 CPU with a clock speed of 2.60GHz. A time limit of one CPU hour was imposed on each run. Table 4 shows the sizes of each instance in the test suite, as well as various characteristics of the solution.  $z_{\text{root}}$  is the value of the relaxation of the root node of the branch-and-bound tree,  $(z_L, z_U)$  are the best lower and upper bounds found in one hour of CPU time, # Nodes is the number of nodes in the enumeration tree, and  $T$  is the CPU seconds on an Intel Pentium 4 CPU with a clock speed of 2.60GHz.

**Table 4.** Impact of Perspective Reformulation on Network Design Instances

Instance	N	K	A	Original Form.				Perspective Form.			
				$z_{\text{root}}$	$(z_L, z_U)$	# Nodes	$T$	$z_{\text{root}}$	$(z_L, z_U)$	# Nodes	$T$
ATL	15	15	22	40.7	(55.4,55.4)	752	116.9	48.3	(55.4,55.4)	464	66.8
R1	20	20	44	37.7	(135.9,172.2)	2488	3600	78.8	(147.5,158.4)	5781	3600
R2	30	30	108	46.8	(140.8,326.9)	253	3600	59.9	(201.5, $\infty$ )	394	3600

For the network design problems, the perspective formulation is always quite useful for improving the lower bounds, and in two of the cases, this translates into improved performance. For the instance R2, Mosek was unable to find a feasible solution to the instance when reformulated via the perspective transformation.

## 5 Conclusions

In this work we derive an explicit characterization of the convex hull of the union of a point and a bounded convex set defined by analytic functions. This characterization can be used to produce strong “perspective” reformulations of many practical mixed integer nonlinear programs. We also show that in many cases, the nonlinear inequalities in the perspective reformulation can be cast as second-order cone constraints, a transformation that greatly improves an instance’s solvability. Computational results on two practical applications show the power of the proposed techniques—in one case solving instances multiple orders of magnitude faster than reported in the literature. Continuing work has two primary thrusts: (1) Automatic detection of structures to which the perspective transformation can be applied; and (2) Studying additional simple structures occurring in practical MINLPs in the hope of deriving strong relaxations.

## Acknowledgement

Author Linderoth would like to acknowledge support from the US Department of Energy under grant DE-FG02-05ER25694, and by IBM, through the faculty partnership program. The support of Mosek, ApS for donating licenses for their software is also greatly appreciated.

## References

1. Günlük, O., Lee, J., Weismantel, R.: MINLP strengthening for separable convex quadratic transportation-cost ufl. Technical Report RC24213 (W0703-042), IBM Research Division (March 2007)
2. Boorstyn, R., Frank, H.: Large-scale network topological optimization. *IEEE Transactions on Communications* 25, 29–47 (1977)
3. Perold, A.F.: Large-scale portfolio optimization. *Management Science* 30, 1143–1160 (1984)
4. Bienstock, D.: Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming* 74, 121–140 (1996)
5. Jobst, N.J., Horniman, M.D., Lucas, C.A., Mitra, G.: Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quantitative Finance* 1, 489–501 (2001)
6. Aktürk, S., Atamtürk, A., Gürel, S.: A strong conic quadratic reformulation for machine-job assignment with controllable processing times. Technical Report BCOL Research Report 07.01, Industrial Engineering & Operations Research, University of California, Berkeley (2007)
7. Stubbs, R., Mehrotra, S.: A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming* 86, 515–532 (1999)
8. Balas, E., Ceria, S., Cornuejols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming* 58, 295–324 (1993)
9. Cezik, M.T., Iyengar, G.: Cuts for mixed 0-1 conic programming. *Mathematical Programming* 104, 179–202 (2005)

10. Gomory, R.E.: An algorithm for the mixed integer problem. Technical Report RM-2597, The RAND Corporation (1960)
11. Atamtürk, A., Narayanan, V.: Conic mixed integer rounding cuts. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, Springer, Heidelberg (2007)
12. Nemhauser, G., Wolsey, L.: A recursive procedure for generating all cuts for 0-1 mixed integer programs. *Mathematical Programming* 46, 379–390 (1990)
13. Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0-1 mixed integer programs. *Mathematical Programming* 106, 225–236 (2006)
14. Stubbs, R.A.: Branch-and-Cut Methods for Mixed 0-1 Convex Programming. PhD thesis, Northwestern University (December 1996)
15. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software* 11-12, 625–653 (1999)
16. Mosek: Mosek ApS (2004), <http://www.mosek.com>
17. Ben-Tal, A., Nemirovski, A.: *Lectures on Modern Convex Optimization*. SIAM, Philadelphia (2001)
18. Ceria, S., Soares, J.: Convex programming for disjunctive optimization. *Mathematical Programming* 86, 595–614 (1999)
19. Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A.: An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization* (to appear)
20. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming* 106(1), 25–57 (2006)
21. Lee, J.: Mixed-integer nonlinear programming: Some modeling and solution issues. *IBM Journal of Research & Development* 51, 489–497 (2007)
22. Bertsekas, D., Gallager, R.: *Data Networks*. Prentice-Hall, Englewood Cliffs (1987)
23. Borchers, B., Mitchell, J.E.: An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research* 21, 359–368 (1994)
24. Orłowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0—survivable network design library (2007) *Optimization Online Preprint*, [http://www.optimization-online.org/DB\\_FILE/2007/08/1746.pdf](http://www.optimization-online.org/DB_FILE/2007/08/1746.pdf)

# Disjunctive Cuts for Non-convex Mixed Integer Quadratically Constrained Programs

Anureet Saxena<sup>1</sup>, Pierre Bonami<sup>2</sup>, and Jon Lee<sup>3</sup>

<sup>1</sup> Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213, USA  
anureets@andrew.cmu.edu

<sup>2</sup> Laboratoire d'Informatique Fondamentale de Marseille,  
CNRS-Université de Marseille, France  
pierre.bonami@lif.univ-mrs.fr

<sup>3</sup> IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 USA  
jonlee@us.ibm.com

**Abstract.** This paper addresses the problem of generating strong convex relaxations of Mixed Integer Quadratically Constrained Programming (MIQCP) problems. MIQCP problems are very difficult because they combine two kinds of non-convexities: integer variables and non-convex quadratic constraints. To produce strong relaxations of MIQCP problems, we use techniques from disjunctive programming and the lift-and-project methodology. In particular, we propose new methods for generating valid inequalities by using the equation  $Y = xx^T$ . We use the concave constraint  $0 \succcurlyeq Y - xx^T$  to derive disjunctions of two types. The first ones are directly derived from the eigenvectors of the matrix  $Y - xx^T$  with positive eigenvalues, the second type of disjunctions are obtained by combining several eigenvectors in order to minimize the *width* of the disjunction. We also use the convex SDP constraint  $Y - xx^T \succcurlyeq 0$  to derive convex quadratic cuts and combine both approaches in a cutting plane algorithm. We present preliminary computational results to illustrate our findings.

## 1 Introduction

In this paper we study the mixed integer quadratically constrained program defined as follows:

$$\begin{aligned} \min \quad & a_0^T x \\ \text{subject to} \quad & x^T A_i x + a_i^T x + b_i \leq 0, \quad i = 1 \dots m; \\ & x_j \in \mathbb{Z}, \quad j \in N_I; \\ & l \leq x \leq u, \end{aligned} \tag{MIQCP'}$$

where  $N$  ( $n = |N|$ ) denotes the set of variables,  $N_I$  denotes the set of integer constrained variables,  $A_i$  ( $i = 1 \dots m$ ) are  $n \times n$  symmetric (usually not positive semidefinite) matrices,  $a_i$  ( $i = 0 \dots m$ ),  $l$  and  $u$  are  $n$ -dimensional vectors and  $b_i$  ( $i = 1 \dots m$ ) are scalars. The decision variant of **MIQCP'** is well known

to be undecidable, even in the pure integer case, when the variables are not bounded (see [10]). Many natural applications of **MIQCP'** can be found in the global-optimization literature. In this paper our focus is to derive tight convex relaxations for **MIQCP'** by using cutting plane approaches.

A standard approach to derive a convex relaxation of **MIQCP'** is to first introduce extra variables  $Y_{ij} = x_i x_j$  in the formulation. Consequently, the following lifted reformulation of **MIQCP'** is obtained

$$\begin{aligned}
 \min \quad & a_0^T x \\
 \text{s.t.} \quad & A_i \cdot Y + a_i^T x + b_i \leq 0, \quad i = 1 \dots m; \\
 & x_j \in \mathbb{Z}, \quad j \in N_I; \\
 & l \leq x \leq u; \\
 & Y = xx^T.
 \end{aligned} \tag{MIQCP}$$

Note that the only non-convex constraint in **MIQCP** is the set of non-linear equations  $Y = xx^T$ , which can be relaxed as a pair of SDP inequalities  $Y - xx^T \succcurlyeq 0$  and  $xx^T - Y \succcurlyeq 0$ . The former of these inequalities can be expressed as a LMI (Linear Matrix Inequality) on the cone of positive semi-definite matrices, while the latter non-convex inequality constitutes the topic of this paper.

One of the common predicaments for non-convex problems is that they are composed of seemingly innocuous looking non-convex constraints (for example  $x_i \in \{0, 1\}$ ) linked together through a set of (usually linear or convex) constraints. For instance, a mixed integer 0-1 program is composed of a linear program and binary  $\{0, 1\}$  constraints on some of the variables. In these kind of problems, convexifying the non-convex constraints does not yield any significant improvement until the convexification process explicitly takes into account the constraints linking the non-convexities together. For instance, convexifying the 0-1 conditions on a set of binary variables in a MILP yields the unit hypercube, which obviously offers little help in solving an MILP.

Interestingly, most of the existing convexification based approaches in MINLP fail to take the linking constraints into account and work exclusively with non-convex sets, and try to derive closed form expressions for the convexified sets [23, 24]. Some other approaches try to perform local convexification and impose that by additional constraints. For instance, imposing the SDP constraint  $Y - xx^T \succcurlyeq 0$  falls into this category of approaches. Naturally, we are interested in an approach which takes a holistic view of the problem and tries to capitalize on the interaction between the problem constraints. In this paper, we propose to use the framework of disjunctive programming to accomplish this goal.

Classical disjunctive programming of Balas [2] requires a linear relaxation of the problem and a disjunction that is satisfied by all the feasible solution to the problem. As is now customary in the MINLP literature, we will use the outer-approximation (OA) of **MIQCP** as the quintessential linear relaxation. We use the phrase “suitably defined OA” in this paper to emphasize the dependence of OA under discussion on the incumbent solution  $(\hat{x}, \hat{Y})$  to the convex relaxation of **MIQCP**.



As for the choice of disjunctions, we seek the sources of non-convexities in **MIQCP**. Evidently, **MIQCP** has two of these, namely, the integrality conditions on the  $x_j$  ( $j \in N_I$ ) variables and the non-linear equations  $Y = xx^T$ . Integrality constraints have been used to derive disjunctions in MILP for the past five decades, and we do not add anything new to this body of work. Our main contribution lies in deriving valid disjunctions from  $Y = xx^T$ , by analyzing the eigenvectors of the matrix  $\hat{Y} - \hat{x}\hat{x}^T$ , and deriving univariate expressions of the form  $Y.c c^T \leq (c^T x)^2$  which are subsequently used to derive disjunctive cuts.

The rest of the paper is organized as follows. In §2 we revisit some of the basic ideas in disjunctive programming and give a detailed description of our disjunctive cut generator. In §3 we derive a large class of valid disjunctions for **MIQCP** and establish interesting connections with elementary 0-1 disjunctions in MILP. §4 investigates the problem of designing disjunctions that use more problem information than is available from the eigenvectors of  $\hat{Y} - \hat{x}\hat{x}^T$ . We introduce the notion of the *width* of a disjunction, and show that disjunctions with smaller widths are likely to give rise to stronger disjunctive cuts. We build on this observation and design a MILP model to find better disjunctions. A scheme for diversifying the class of disjunctions based on the Gram-Schmidt orthogonalization procedure is briefly discussed. Finally, in §5 we report preliminary computational results on three types of instances: selected problems from GLOBALLib [11], some examples of **MIQCP** instances from [12] which arise in chemical engineering applications and some continuous boxed constrained QPs from [26].

## 2 Disjunctive Programming

In this section we review some of the basic ideas from disjunctive programming and give a detailed description of our cut generator. Given a polytope  $P = \{x \mid Ax \geq b\}$ , a disjunction  $D = \bigvee_{t=1}^q (D^t x \geq d^t)$  and a point  $\hat{x} \in P$ , a central question in disjunctive programming is to show that  $\hat{x} \in Q = \text{clconv} \bigcup_{t=1}^q \{x \in P \mid D^t x \geq d^t\}$  or find a valid inequality  $\alpha x \geq \beta$  for  $Q$  that is violated by  $\hat{x}$ .

This question arises in several areas of computational optimization where the specific form of the polytope  $P$  and disjunction  $D$  is governed by the underlying application. For instance, in the context of mixed integer linear programming (MILP), the polytope  $P$  usually represents the LP-relaxation of the MILP, while the disjunctions are obtained by exploiting the integrality constraints (see for example [3,4]). Similarly, in the context of probabilistic programming,  $P$  usually represents the deterministic variant of the problem, whereas the disjunction is derived from the so-called  $p$ -efficient frontier [19,18]. In the context of **MIQCP**,  $P$  will represent a suitably chosen outer-approximation of **MIQCP**, while the disjunction is obtained by exploiting the integrality constraints on the variables  $x_j$  ( $j \in N_I$ ) or from the eigenvectors of the matrix  $Y - xx^T$  (see §3).

The theorem that follows formulates the separation problem mentioned above as a linear program. It follows immediately from the results presented in [2].

**Theorem 1.**  $\hat{x} \in Q$  if and only if the following linear program is infeasible:

$$\begin{aligned}
\min \quad & \alpha \hat{x} - \beta \\
\text{s.t.} \quad & \alpha = u^t A + v^t D^t, \quad t = 1 \dots q; \\
& \beta \leq u^t b + v^t d^t, \quad t = 1 \dots q; \\
& u^t, v^t \geq 0, \quad t = 1 \dots q; \\
& \sum_{t=1}^q (u^t \xi + v^t \xi^t) = 1,
\end{aligned} \tag{CGLP}$$

$$\begin{aligned}
& \xi, \xi^t \quad (t = 1 \dots q) \\
& \xi^t > 0 \quad (t = 1 \dots q) \\
& (\alpha, \beta, u^1, v^1, \dots, u^q, v^q) \\
& \alpha x \geq \beta \\
& \text{ff } \hat{x}
\end{aligned}$$

The constraint  $\sum_{t=1}^q (u^t \xi + v^t \xi^t) = 1$  of the CGLP, referred to as the normalization constraint, plays a central role in determining the strength and numerical stability of the resulting cut [4]. In our computational results, we used the following normalization:

1.  $\xi_i^t = 1, \forall i = 1 \dots m_t, t = 1 \dots q$ , where  $m_t$  denotes the number of rows in the matrix  $D_t$ .
- 2.

$$\xi_i = \begin{cases} 0, & \text{for } i \in L; \\ \|a_i\|_1, & \text{otherwise,} \end{cases}$$

where  $L$  denotes the set of lower-bound constraints in  $Ax \geq b$ , while  $a_i$  denotes the  $i^{\text{th}}$  row of the matrix  $A$ .

The above normalization has two important characteristics. First, it implicitly scales the constraints in  $Ax \geq b$  (other than the lower-bound constraints) so that all of them have a  $\ell_1$ -norm of 1, which in turn significantly improves the numerical properties of the resulting cut. Second, assigning a normalization coefficient of zero to the lower-bound constraints allows us to handle these constraints as bounds on variables associated with the dual of the CGLP, thereby speeding up the overall algorithm to solve the CGLP.

In order to use the machinery of disjunctive programming to strengthen the formulation of **MIQCP**, we need a class of disjunctions that are satisfied by every feasible solution to **MIQCP**. Note that **MIQCP** has two sources of non-convexities, namely the integrality constraints on  $x_j$  ( $j \in N_I$ ) variables, and the equality constraints  $Y = xx^T$ . While the former can be used to derive split disjunctions, as is usually done in MILP, the latter need to be handled more carefully. The section that follows gives a novel way of deriving valid disjunctions from the constraints  $Y = xx^T$ .

### 3 Valid Disjunctions for MIQCP

Note that for  $c \in \mathbb{R}^n$ , any feasible solution to **MIQCP** satisfies  $(c^T x)^2 = Y.c c^T$ , which in turn is equivalent to the following two inequalities  $(c^T x)^2 \leq Y.c c^T$  and

$(c^T x)^2 \geq Y.c.c^T$ . The former of these two inequalities is a convex quadratic constraint that can be readily added to the formulation. The second constraint  $(c^T x)^2 \geq Y.c.c^T$ , on the other hand, gives rise to the following disjunction which is satisfied by every feasible solution to **MIQCP**:

$$\left[ \begin{array}{c} \eta_L(c) \leq c^T x \leq \theta \\ -(c^T x)(\eta_L(c) + \theta) + \theta\eta_L(c) \leq -Y.c.c^T \end{array} \right] \vee \left[ \begin{array}{c} \theta \leq c^T x \leq \eta_U(c) \\ -(c^T x)(\eta_U(c) + \theta) + \theta\eta_U(c) \leq -Y.c.c^T \end{array} \right], \quad (1)$$

where  $\eta_L(c) = \min\{c^T x \mid (x, Y) \in \tilde{P}\}$ ,  $\eta_U(c) = \max\{c^T x \mid (x, Y) \in \tilde{P}\}$ ,  $\tilde{P}$  is a suitably chosen relaxation of **MIQCP** and  $\theta \in (\eta_L(c), \eta_U(c))$ . In our computational experiments, we chose  $\tilde{P}$  to be a suitably defined outer-approximation of **MIQCP**, and  $\theta = \frac{\eta_L(c) + \eta_U(c)}{2}$ . The above disjunction can be derived by splitting the range  $[\eta_L(c), \eta_U(c)]$  of the function  $c^T x$  over  $\tilde{P}$  into two intervals  $[\eta_L(c), \theta]$  and  $[\theta, \eta_U(c)]$ , and constructing a secant approximation of the function  $-(c^T x)^2$  in each of the intervals, respectively. The above disjunction can then be used to derive disjunctive cuts by using the apparatus of CGLP. Furthermore, for any integer  $q > 1$  a  $q$ -term disjunction can be obtained by splitting the  $[\eta_L(c), \eta_U(c)]$  interval into  $q$  parts, and constructing a secant approximation of  $-(c^T x)^2$  in each one of the  $q$  intervals. Non-convex inequalities of the form  $(c^T x)^2 \geq Y.c.c^T$  are referred to as *MIQCP cuts* in the sequel.

From a computational standpoint, the only question that remains to be answered is, how can we judiciously choose a vector  $c$  that is likely to give rise to strong cuts. We describe two procedures for deriving such vectors; both of these procedures use the eigenvectors of the matrix  $\hat{Z} = \hat{Y} - \hat{x}\hat{x}^T$  where  $(\hat{x}, \hat{Y})$  denotes the incumbent solution to the convex relaxation of **MIQCP** that we want to cut off. Let  $c_1, \dots, c_n$  denote a set of orthonormal eigenvectors of  $\hat{Z}$ , and let  $\mu_1 \geq \mu_2 \geq \dots \geq \mu_n$  be the corresponding eigenvalues.

Let  $k \in \{1, \dots, n\}$  and let  $c = c_k$ . Note that if  $\mu_k < 0$ , then  $(c^T x)^2 \leq Y.c.c^T$  is a valid convex quadratic cut which cuts off  $(\hat{x}, \hat{Y})$ . If  $\mu_k > 0$ , then  $(c^T x)^2 \geq Y.c.c^T$  is a valid inequality (albeit non-convex) for **MIQCP** which cuts off  $(\hat{x}, \hat{Y})$ . Consequently, in this case the disjunction derived from  $(c^T x)^2 \geq Y.c.c^T$  is a good candidate for generating disjunctive cuts. In our computational experiments, we added a convex quadratic cut for every negative eigenvalue of  $\hat{Z}$ , and generated a disjunctive cut (if any) from every positive eigenvalue of  $\hat{Z}$ .

Several comments are in order. First, the relaxation of **MIQCP** obtained by replacing  $Y = xx^T$  by  $Y - xx^T \succcurlyeq 0$  has been studied by several other authors ([13], [21], [6], [1]). From engineering viewpoint, incorporating the positive semi-definiteness condition  $Y - xx^T \succcurlyeq 0$  as part of the relaxation poses a serious hurdle, since most general purpose solvers for nonlinear optimization (such as Ipopt [27], FilterSQP [9]) are not designed to handle conic constraints of the form  $Y - xx^T \succcurlyeq 0$  (or equivalently  $\begin{bmatrix} 1 & x \\ x^T & Y \end{bmatrix} \succcurlyeq 0$ ). Special purpose softwares for conic programming (such as SeDuMi [22]), on the other hand, cannot handle

arbitrary convex constraints. Since our solver for the convex relaxations Ipopt [27] is a general purpose solver, we incorporated the effect of  $Y - xx^T \succeq 0$  by iteratively generating convex quadratic inequalities  $(c^T x)^2 \leq Y.c.c^T$  derived from eigenvectors  $c$  of  $\hat{Z}$  associated with negative eigenvalues.

Second, our approach of strengthening the relaxation of **MIQCP** by generating disjunctive cuts can also be viewed as convexifying the feasible region of **MIQCP**. Convexification of non-convex feasible regions is an active research area in MINLP community ([23,24,25,26]). Most of these convexification based approaches, however, aim to convexify non-convex problem constraints individually, and often fail to exploit the interaction across problem constraints to derive stronger cuts. A disjunctive programming based approach, such as the one presented in this paper, takes a holistic view of the problem and tries to draw stronger inferences from disjunctive cuts by combining information from all the problem constraints.

Third, there is an interesting connection between univariate expressions  $(c^T x)^2 \geq Y.c.c^T$  derived from eigenvectors  $c$  of  $\hat{Z}$ , and elementary 0-1 disjunctions encountered in MILP. For the sake of discussion, consider a mixed 0-1 program  $\min\{\tilde{c}z \mid \tilde{A}z \geq \tilde{b}, z_j \in \{0, 1\} \forall j \in J\}$ , and let  $\tilde{z}$  be the optimal solution to the LP-relaxation of this problem. Recall that the fractionality (or integer infeasibility) of  $\tilde{z}_j$  ( $j \in J$ ) is defined to be  $\min\{\tilde{z}_j, 1 - \tilde{z}_j\}$ . Elementary 0-1 disjunctions associated with components of  $\tilde{z}$  with higher infeasibility are often preferred over others for generating disjunctive cuts. With respect to **MIQCP**, the disjunctive cut of  $(\hat{x}, \hat{Y})$  with respect to the univariate expression  $(c^T x)^2 \geq Y.c.c^T$  can be defined to be  $\frac{c^T(\hat{Y} - \hat{x}\hat{x}^T)c}{\|c\|_2}$ . Consequently, the problem of choosing a univariate expression with maximum infeasibility can be phrased as,  $\max_{\|c\|_2=1} c^T(\hat{Y} - \hat{x}\hat{x}^T)c$ ; clearly  $c_1$  is the optimal solution to this problem with optimal solution value of  $\mu_1$ . To summarize, our choice of using the eigenvectors of  $\hat{Z}$  to construct univariate expressions is akin to choosing the most fractional variable for generating a lift-and-project cut in the MILP framework.

## 4 More Disjunctions

Note that univariate expressions derived from eigenvectors of  $\hat{Z}$  are oblivious to other constraints in the problems. In other words, these eigenvectors are not influenced by majority of the problem constraints, and hence do not completely exploit the problem structure. In this section, we give a systematic procedure for generating univariate expressions that utilize all the problem constraints, and are hence likely to give rise to stronger cuts (also see [5]).

For  $c \in \mathbb{R}^n$  let  $\eta_L(c) = \min\{c^T x \mid (x, Y) \in P\}$  and  $\eta_U(c) = \max\{c^T x \mid (x, Y) \in P\}$  for a suitably chosen outer approximation  $P$  of **MIQCP**. Let  $\eta(c) = \eta_U(c) - \eta_L(c)$  denote the width of the interval  $[\eta_L(c), \eta_U(c)]$ . The following inequality represents the secant approximation of the function  $-(c^T x)^2$  in the  $[\eta_L(c), \eta_U(c)]$  interval, and is hence a valid disjunctive cut derived from the disjunction (II).

$$-c^T x(\eta_L(c) + \eta_U(c)) + \eta_L(c)\eta_U(c) \leq -Y.c.c^T. \quad (2)$$

The proposition that follows gives a closed form expression for the maximum error incurred by the secant approximation of the negative square function in a bounded interval.

**Proposition 1.**  $f : \mathbb{R} \rightarrow \mathbb{R}$ ,  $f(x) = -x^2$ ,  $g(x) = -x(a + b) + ab$ ,  $f(x) \geq g(x)$   $\forall x \in [a, b]$  ( $a, b \in \mathbb{R}$ )  
 $\max_{x \in [a, b]} (f(x) - g(x)) = \frac{(a-b)^2}{4}$

As a direct consequence of the above proposition it follows that secant approximation error incurred by (2) is proportional to  $\eta(c)^2$ . Consequently, we can use  $-\eta(c)$  as a metric to measure the strength of the disjunctive cuts obtainable from  $Y.c.c^T \leq (c^T x)^2$ . The proposition that follows show that  $\eta(c)$  can be computed by solving a linear program.

**Proposition 2.**  $P = \{(x, Y) \mid Ax + BY \geq b\}$ ,  $B \succcurlyeq 0$

$$\eta(c) = - \max (u + v)^T b$$

$$\begin{aligned} uA &= c; \\ -vA &= c; \\ u.B &= 0; \\ v.B &= 0; \\ u, v &\geq 0. \end{aligned}$$

To summarize, we are looking for vectors  $c \in \mathbb{R}^n$  whose univariate expression  $Y.c.c^T \leq (c^T x)^2$  is violated by  $(\hat{x}, \hat{Y})$  and has a small width  $\eta(c)$ . Note that if we restrict our attention to the subspace spanned by eigenvectors of  $\hat{Z}$  with positive eigenvalues, then the first condition is automatically satisfied. Thus, we can model the problem of determining a vector  $c$  that gives the best univariate expression as

$$\min \eta(c) - \epsilon \left( \sum_{j=1}^n |\lambda_j| \mu_j \right)$$

$$\begin{aligned} c &= \sum_{j=1}^n \lambda_j c_j \\ \sum_{j=1}^n |\lambda_j| &= 1; \\ \lambda_j &= 0, \quad \forall j \in \{1 \dots n\} \text{ s.t. } \mu_j \leq 0. \end{aligned}$$

In the above model, the constraint  $\sum_{j=1}^n |\lambda_j| = 1$  models that the  $\ell_1$ -norm of  $c$  expressed in the basis defined by  $(c_1, \dots, c_n)$  is equal to 1. The constraint  $\lambda_j = 0 \forall j$  s.t  $\mu_j \leq 0$  ensures that  $c$  lies in the subspace spanned by eigenvectors of  $\hat{Z}$  with positive eigenvalues. The penalty term  $\epsilon \left( \sum_{j=1}^n |\lambda_j| \mu_j \right)$  ( $\epsilon = 10^{-4}$ ) expresses our desire to bias the  $c$  vector towards eigenvectors with large eigenvalues. The above model can be easily recast as the following mixed integer program, referred to as Univariate-expression Generating Mixed Integer Program

$$\min -(u + v)^T b - \sum_{j=1}^n \lambda_j^+ \mu_j \epsilon$$

$$\begin{aligned} & uA = c ; \\ & -vA = c ; \\ & u.B = 0 ; \\ & v.B = 0 ; \\ & u, v \geq 0 ; \\ & c = \sum_{j=1}^n \lambda_j c_j ; \\ & z_j - 1 \leq \lambda_j, \forall j = 1 \dots n ; \\ & \lambda_j \leq z_j, \forall j = 1 \dots n ; \\ & \lambda_j^+ \leq \lambda_j + 2(1 - z_j), \forall j = 1 \dots n ; \\ & \lambda_j^+ \leq -\lambda_j + 2z_j, \forall j = 1 \dots n ; \\ & \lambda_j^+ \geq 0, \forall j = 1 \dots n ; \\ & \sum_{j=1}^n \lambda_j^+ = 1 ; \\ & \lambda_j = 0, \forall j \in \{1 \dots n\} \text{ s.t. } \mu_j \leq 0 ; \\ & z_j \in \{0, 1\} \quad j \in \{1, \dots, n\}. \end{aligned} \tag{UGMIP}$$

Another idea that has played a significant role in the successful application of general-purpose cutting planes in MILP is that of [\[84\]](#). Cut diversification refers to the strategy of adding a batch of cuts each of which affects a different part of the incumbent solution thereby triggering a collaborative action and yielding improvements that cannot be obtained by a single cut. For instance, the tremendous practical performance of Mixed Integer Gomory Cuts is often attributed to their well-diversified nature (see [\[7\]](#)). Interestingly, the above UGMIP can be easily augmented to generate a set of diversified vectors  $c$  instead of a single vector. To see this, suppose a set of vectors  $c^k = \sum_{j=1}^n \lambda_j^{(k)} c_j$  ( $i = 1 \dots K$ ) has already been generated, and we are interested in finding a vector  $c$  that is different from  $c^k$  ( $k = 1 \dots K$ ). This can be accomplished by amending the UGMIP by appending the following constraints which not only exclude the vectors  $c^k$  ( $k = 1 \dots K$ ), but also ensure mutual orthogonality between any feasible solution of UGMIP and  $c^k$  ( $k = 1 \dots K$ ):

$$\sum_{j=1}^n \lambda_j \lambda_j^k = 0 \quad \forall k = 1 \dots K .$$

In our computational experiments we solved UGMIP using CPLEX 10.1 enumerating at most 2000 branch-and-bound nodes. Furthermore, the diversification scheme mentioned above was used iteratively until the resulting UGMIP became infeasible or CPLEX was unable to find a feasible solution within the stipulated node limit.

## 5 Computational Results

We report preliminary computational results in this section. Since the aim of these experiments was to assess the performance of different classes of cutting planes and their relative strengths, we report the percentage duality gap closed by each one of them at the root node. All of the experiments described in this section used the following general setup:

1. Solve the convex relaxation of **MIQCP**.
2. Generate cutting planes to cut off  $(\hat{x}, \hat{Y})$ .
3. If a violated cut was generated, then goto step (II), else STOP.

The above loop was repeated until a time-limit of 60 minutes was reached or the code was unable to find any violated cut. We implemented the following three variants of cutting planes discussed in the previous sections.

- **Variante 1:** Only convex quadratic cuts derived from eigenvectors associated with negative eigenvalues of  $\hat{Y} - \hat{x}\hat{x}^T$  were used.
- **Variante 2:** Same as Variante 1, except that disjunctive cuts from univariate expression derived from eigenvectors of  $\hat{Z}$  with positive eigenvalues were also used.
- **Variante 3:** Same as Variante 2 except that disjunctive cuts from additional univariate expressions found by using the UGMIP machinery and diversification scheme were also used.

The three variants were implemented using the open-source framework Bonmin [5] from COIN-OR. The nonlinear solver used is Ipopt [27], the eigenvalue problems are solved using Lapack and the cut generation linear programs are solved using CPLEX10.1. Two comments are in order. First, we strengthen the initial convex relaxation of **MIQCP** by adding the following RLT inequalities [14,20], for  $i, j \in \{1 \dots n\}$  such that  $i \leq j$ ,

$$\begin{aligned}
 y_{ij} - l_i x_j - u_j x_i + l_i u_j &\leq 0 \\
 y_{ij} - l_j x_i - u_i x_j + l_j u_i &\leq 0 \\
 y_{ij} - l_j x_i - l_i x_j + l_j l_i &\geq 0 \\
 y_{ij} - u_j x_i - u_i x_j + u_j u_i &\geq 0.
 \end{aligned}$$

Second, while generating the disjunctive cuts we remove all the RLT inequalities from the outer approximation except those which are binding at the incumbent solution. While solving the CGLP we use a column generation based approach to generate  $u_i^t$  variables corresponding to non-binding RLT inequalities. Since there is a huge number  $O(n^2)$  of RLT inequalities, we found it more efficient to use a column generation based approach to handle them while solving the CGLPs thereby exploiting the reoptimization capabilities of the CPLEX linear programming solver. Since Ipopt has a very limited support for warm-starting, we found it more suitable to supply all the RLT inequalities simultaneously while solving the convex relaxations.

Next we describe our computational results on the following three test-beds: GLOBALLib [11], instances from Lee and Grossmann [12] and Box-QP instances.

GLOBALLib is a repository of 413 global optimization instances of widely varying types and sizes. Of these 413 instances, we selected all problems with at most 50 variables which can be easily converted into instances of **MIQCP**. For instance, some of the problems have polynomial terms  $(x_1 x_2 x_3 x_4 x_5, x_1^3$  etc) which can be converted into quadratic expressions by introducing additional variables. Similarly, some of the problems do not have explicit upper bounds on

**Table 1.** Summary Results: GLOBALlib instances with non-zero Duality Gap

	V1	V2	V3
>99.99 % gap closed	16	23	23
98-99.99 % gap closed	1	44	52
75-98 % gap closed	10	23	21
25-75 % gap closed	11	22	20
0-25 % gap closed	91	17	13
Total Number of Instances	129	129	129
Average Gap Closed	24.80%	76.49%	80.86%

**Table 2.** Summary of results on the Lee-Grossmann examples

Instance	RLT	Opt	V1	V2	V3
Example 1	-58.70	-11	-58.70	-37.44	-37.44
Example 2	-414.94	-14	-93.19	-14.26	-14.26
Example 3	-819.66	-510.08	-793.15	-513.61	- 511.10
Example 4	-499282.59	-116,575	-472,727.49	-363,487.69	-359,618.10

the variables; for such problems we used linear programming techniques to determine valid upper bounds thereby making them amenable to techniques discussed in this paper. The final set of selected problems comprised 160 instances<sup>1</sup>

We found that Ipopt encounters numerical problems on 6 of the selected instances, which were later excluded from our experiments. For one instance (alkylation) there is no known feasible solution which makes it difficult to assess the performance of cutting planes. Out of the remaining 153 instances, 24 instances have zero duality gap<sup>2</sup>; in other words the RLT relaxation already closes 100% of the gap on these instances. Tables 3, 4 and 5 report the computational results on the remaining 129 instances, while Table 1 reports the same in summarized form. The second column of Tables 3, 4 and 5 reports the optimal value of the RLT relaxation of **MIQCP**, while the third column reports the value of the best known solution. Note that either variant 2 or variant 3 closes more than 99% of the duality gap on some of the instances (st\_qpc-m3a, st\_ph13, st\_ph11, ex3\_1\_4, st\_jcbpaf2, ex2\_1\_9 etc) on which variant 1 is unable to close any gap. Furthermore, variant 3 closes 10% more duality gap than variant 2 on some of the instances (ex2\_1\_1, ex3\_1\_4, ex5\_2\_4, ex7\_3\_1, ex9\_2\_3, st\_pan2 etc) showing the interest of disjunctions obtained from solution of the UGMIP problem.

Finally, in order to assess the performance of our code on the 24 instances with no duality gap, we report the spectral norm of  $\hat{Y} - \hat{x}\hat{x}^T$  in Table 6, where  $(\hat{x}, \hat{Y})$  denotes the incumbent solution at the last iteration of the respective variant.

<sup>1</sup> Which can be downloaded in AMPL .mod format from [www.andrew.cmu.edu/user/anureets/MIQCP](http://www.andrew.cmu.edu/user/anureets/MIQCP)

<sup>2</sup> We define the duality gap closed by a relaxation  $\mathcal{I}$  of **MIQCP** as,  $\frac{\text{opt}(\mathcal{I}) - \text{RLT}}{\text{opt} - \text{RLT}} \times 100$ , where  $\text{opt}(\mathcal{I})$ ,  $\text{RLT}$ , and  $\text{opt}$  denote the optimal value of  $\mathcal{I}$ , the RLT relaxation of **MIQCP** and **MIQCP**, respectively.



**Table 3.** GLOBALLib Instances with non-zero Duality Gap (Part 1)

Instance	RLT	OPT	% Duality Gap Closed			Time(sec)		
			V1	V2	V3	V1	V2	V3
alkyl	-2.7634	-1.7650	0.00	55.83	63.75	10.621	3619.874	3693.810
circle	0.0000	4.5742	45.74	99.89	99.84	0.218	0.456	0.664
dispatch	3101.2805	3155.2879	100.00	100.00	100.00	0.044	0.052	0.066
ex2_1_1	-18.9000	-17.0000	0.00	72.62	99.92	0.009	704.400	17.835
ex2_1_10	39668.0556	49318.0180	22.05	99.37	99.82	6.719	29.980	70.168
ex2_1_5	-269.4528	-268.0146	0.00	99.98	99.99	0.020	0.173	0.188
ex2_1_6	-44.4000	-39.0000	0.00	99.95	99.97	0.023	3397.650	54.326
ex2_1_7	-6031.9026	-4150.4101	0.00	41.17	45.58	0.188	3607.439	3763.506
ex2_1_8	-82460.0000	15639.0000	0.00	84.70	92.75	0.491	3632.275	3627.700
ex2_1_9	-2.2000	-0.3750	0.00	98.79	99.73	0.140	1587.940	3615.766
ex3_1_1	2533.2008	7049.2480	0.00	15.94	22.13	1.391	3600.268	3681.021
ex3_1_2	-30802.7563	-30665.5387	49.74	99.99	99.99	0.035	0.083	0.108
ex3_1_3	-440.0000	-310.0000	0.00	99.99	99.99	0.013	0.064	0.096
ex3_1_4	-6.0000	-4.0000	0.00	86.31	99.57	0.009	21.261	581.295
ex4_1_1	-173688.7998	-7.4873	100.00	100.00	100.00	0.287	0.310	0.444
ex4_1_3	-7999.4583	-443.6717	56.40	93.54	99.86	0.080	0.285	0.552
ex4_1_4	-200.0000	0.0000	100.00	100.00	100.00	0.247	0.243	0.532
ex4_1_6	-24075.0002	7.0000	100.00	100.00	100.00	0.185	0.308	0.508
ex4_1_7	-206.2500	-7.5000	100.00	100.00	100.00	0.128	0.114	0.165
ex4_1_8	-29.0000	-16.7389	100.00	100.00	100.00	0.043	0.059	0.103
ex4_1_9	-6.9867	-5.5080	0.00	43.59	37.48	0.008	1.307	1.273
ex5_2_2_case1	-599.8996	-400.0000	0.00	0.00	0.00	0.011	0.016	0.935
ex5_2_2_case2	-1200.0000	-600.0000	0.00	0.00	0.00	0.021	0.047	0.511
ex5_2_2_case3	-875.0000	-750.0000	0.00	0.36	0.31	0.016	0.358	0.474
ex5_2_4	-2933.3334	-450.0000	0.00	79.31	99.92	0.046	68.927	1044.400
ex5_2_5	-9700.0001	-3500.0001	0.00	6.27	6.37	1.825	3793.169	3618.084
ex5_3_2	0.9979	1.8642	0.00	7.27	21.00	0.355	245.821	3672.529
ex5_3_3	1.6313	3.2340	0.00	0.21	0.18	3764.946	3693.758	7511.839
ex5_4_2	2598.2452	7512.2301	0.00	27.57	26.41	1.141	3614.376	3866.626
ex7_3_1	0.0000	0.3417	0.00	0.00	85.43	0.313	5.582	3622.223
ex7_3_2	0.0000	1.0899	0.00	59.51	70.26	0.788	3609.704	3614.759
ex8_1_3	-7.7486E+12	1.0000	0.04	0.04	0.00	0.509	0.494	0.641
ex8_1_4	-13.0000	0.0000	100.00	100.00	100.00	0.020	0.038	0.051
ex8_1_5	-3.3333	0.0000	68.30	68.97	68.96	0.839	1.246	100.476
ex8_1_7	-757.5775	0.0293	77.43	77.43	95.79	75.203	75.203	3615.517
ex8_1_8	-0.8466	-0.3888	0.00	76.49	90.88	7.722	3607.682	3628.366
ex8_4_1	-5.0000	0.6186	91.84	91.09	86.49	3659.232	3642.131	4180.427
ex8_4_2	-5.0000	0.4852	94.07	93.04	87.87	3641.875	3606.071	3757.098
ex9_1_4	-63.0000	-37.0000	0.00	0.00	1.55	0.077	0.603	244.126
ex9_2_1	-16.0000	17.0000	54.54	60.04	92.02	3603.428	2372.638	3622.960

Note that we were able to generate almost feasible solutions (i.e spectral-norm  $\leq 10^{-4}$ ) on 17 out of 24 instances.

The ex9\* instances in the GLOBALLib repository contain the linear complementarity constraints (LCC)  $x_i x_j = 0$  on a subset of variables. These constraints give rise to the following disjunction,  $(x_i = 0) \vee (x_j = 0)$ , which in turn can be embedded within the CGLP framework to generate disjunctive cuts. In order

**Table 4.** GLOBALLib Instances with non-zero Duality Gap (Part 2)

Instance	RLT	OPT	% Duality Gap Closed			Time(sec)		
			V1	V2	V3	V1	V2	V3
ex9_2.2	-50.0000	100.0000	70.37	88.29	98.06	1227.898	3606.357	3610.411
ex9_2.3	-30.0000	0.0000	0.00	0.00	47.17	0.125	3.819	3625.114
ex9_2.4	-396.0000	0.5000	99.87	99.87	99.89	2.801	8.897	5.258
ex9_2.6	-406.0000	-1.0000	87.23	87.93	62.00	851.127	2619.018	1058.376
ex9_2.7	-9.0000	17.0000	42.31	51.47	86.25	3602.364	3628.249	3627.920
himmel11	-30802.7566	-30665.5387	49.74	99.99	99.99	0.053	0.082	0.120
house	-5230.5433	-4500.0000	0.00	86.93	97.92	0.435	12.873	149.678
hydro	4019717.9291	4366944.1597	100.00	100.00	100.00	8.354	20.668	191.447
mathopt1	-912909.0091	0.0000	100.00	100.00	100.00	1.727	2.448	3.770
mathopt2	-11289.0001	0.0000	100.00	100.00	100.00	0.351	0.229	0.400
meanvar	0.0000	5.2434	100.00	100.00	100.00	0.179	0.276	0.657
nemhaus	0.0000	31.0000	53.97	100.00	100.00	0.836	0.198	0.355
prob05	0.3151	0.7418	0.00	99.78	99.49	0.007	0.165	0.173
prob06	1.0000	1.1771	100.00	100.00	100.00	0.023	0.024	0.031
prob09	-100.0000	0.0000	100.00	99.99	100.00	0.582	0.885	1.689
process	-2756.5935	-1161.3366	7.68	88.05	95.03	6.379	3620.085	3611.299
qp1	-1.4313	0.0008	85.76	89.12	81.23	3659.085	3897.521	3700.918
qp2	-1.4313	0.0008	86.13	89.15	83.06	3643.188	4047.592	4255.863
rbrock	-659984.0066	0.0000	100.00	100.00	100.00	0.353	3.194	5.611
st_bpaf1a	-46.0058	-45.3797	0.00	81.73	88.52	0.049	0.894	3.790
st_bpaf1b	-43.1255	-42.9626	0.00	90.73	92.86	0.047	3.299	12.166
st_bpv2	-11.2500	-8.0000	0.00	99.99	99.99	0.033	0.029	0.034
st_bsj2	-0.6260	1.0000	0.00	99.98	99.96	0.009	1.974	2.235
st_bsj3	-86768.5509	-86768.5500	0.00	0.00	0.00	0.012	0.011	0.011
st_bsj4	-72700.0507	-70262.0500	0.00	99.86	99.80	0.014	1.715	1.384
st_e02	171.4185	201.1591	0.00	99.88	99.95	0.008	0.095	0.118
st_e03	-2381.8947	-1161.3366	29.58	91.63	92.82	715.006	3639.297	3613.883
st_e05	3826.3885	7049.2493	0.00	50.43	58.38	0.194	16.217	41.354
st_e06	0.0000	0.1609	0.00	0.00	0.00	0.215	0.726	1.911
st_e07	-500.0000	-400.0000	0.00	99.97	99.97	0.042	0.350	0.383
st_e08	0.3125	0.7418	0.00	99.81	99.89	0.008	0.208	0.171
st_e09	-0.7500	-0.5000	0.00	92.58	92.58	0.012	0.014	0.018
st_e10	-29.0000	-16.7389	100.00	100.00	100.00	0.036	0.045	0.069
st_e18	-3.0000	-2.8284	100.00	100.00	100.00	0.015	0.018	0.022
st_e19	-879.7500	-86.4222	93.50	95.21	95.18	0.373	0.613	0.991
st_e20	-0.8466	-0.3888	0.00	76.38	90.88	7.409	3610.271	3623.275
st_e23	-3.0000	-1.0833	0.00	98.40	98.40	0.011	0.087	0.108
st_e24	0.0000	3.0000	0.00	99.81	99.81	0.007	0.501	0.657
st_e25	0.2473	0.8902	87.20	100.00	100.00	0.312	0.161	0.247
st_e26	-513.0000	-185.7792	0.00	99.96	99.96	0.006	0.036	0.050

to test the effectiveness of these cuts, we amended our code to automatically detect linear complementarity constraints, and use the corresponding disjunctions along with the default medley of disjunctions to generate disjunctive cuts. Table 7 reports our computational results. It is worth observing that while the default version of our code is unable to close any significant gap on the ex9\_1.4

Table 5. GLOBALLib Instances with non-zero Duality Gap (Part 3)

Instance	RLT	OPT	% Duality Gap Closed			Time(sec)		
			V1	V2	V3	V1	V2	V3
st_e28	-30802.7566	-30665.5387	49.74	99.99	99.99	0.051	0.088	0.118
st_e30	-3.0000	-1.5811	0.00	0.00	0.00	0.014	0.035	6.489
st_e33	-500.0000	-400.0000	0.00	99.94	99.95	0.047	0.457	0.382
st_fp1	-18.9000	-17.0000	0.00	72.62	99.92	0.009	658.824	18.013
st_fp5	-269.4528	-268.0146	0.00	99.98	99.99	0.018	0.175	0.180
st_fp6	-44.4000	-39.0000	0.00	99.92	99.97	0.025	3603.767	54.613
st_fp7a	-435.5237	-354.7506	0.00	45.13	53.58	0.151	806.493	1801.106
st_fp7b	-715.5237	-634.7506	0.00	22.06	55.51	0.153	11.941	3610.617
st_fp7c	-10310.4738	-8695.0122	0.00	44.26	57.10	0.181	3621.180	3672.666
st_fp7d	-195.5237	-114.7506	0.00	50.03	55.53	0.111	3627.749	3734.806
st_fp8	7219.4999	15639.0000	0.00	0.83	3.17	0.331	4.911	88.867
st_glmp_fp2	7.0681	7.3445	0.00	45.70	49.74	0.009	0.732	1.170
st_glmp_kk92	-13.3548	-12.0000	0.00	99.98	99.98	0.023	0.038	0.053
st_glmp_kky	-3.0000	-2.5000	0.00	99.80	99.71	0.011	0.133	0.248
st_glmp_ss1	-38.6667	-24.5714	0.00	89.30	89.30	0.031	0.556	0.736
st_lt	-2.8000	-1.6000	0.00	99.81	99.89	0.006	0.142	0.451
st_iqpbk1	-1722.3760	-621.4878	97.99	99.86	99.99	3.825	5.086	286.844
st_iqpbk2	-3441.9520	-1195.2257	97.93	100.00	100.00	2.515	31.614	243.169
st_jcbpaf2	-945.4511	-794.8559	0.00	99.47	99.61	2.650	3622.733	3636.491
st_jcbpafex	-3.0000	-1.0833	0.00	98.40	98.40	0.012	0.085	0.114
st_kr	-104.0000	-85.0000	0.00	99.93	99.95	0.008	0.090	0.131
st_m1	-505191.3385	-461356.9389	0.00	99.96	99.96	0.222	368.618	756.237
st_m2	-938513.6772	-856648.8187	0.00	70.19	58.99	1.226	3641.449	3876.446
st_pan1	-5.6850	-5.2837	0.00	99.72	99.92	0.007	0.926	0.771
st_pan2	-19.4000	-17.0000	0.00	68.54	99.91	0.009	3038.430	26.401
st_ph1	-243.8112	-230.1173	0.00	99.98	99.98	0.011	0.225	0.059
st_ph11	-11.7500	-11.2813	0.00	99.46	98.19	0.007	0.910	0.337
st_ph12	-23.5000	-22.6250	0.00	99.49	99.62	0.006	0.353	0.311
st_ph13	-11.7500	-11.2813	0.00	99.38	98.80	0.009	0.751	0.703
st_ph14	-231.0000	-229.7222	0.00	99.85	99.86	0.010	0.051	0.131
st_ph15	-434.7346	-392.7037	0.00	99.83	99.81	0.009	0.476	0.541
st_ph2	-1064.4960	-1028.1173	0.00	99.98	99.98	0.014	0.159	0.062
st_ph20	-178.0000	-158.0000	0.00	99.98	99.98	0.007	0.036	0.049
st_ph3	-447.8488	-420.2348	0.00	99.98	99.98	0.011	0.031	0.039
st_phex	-104.0000	-85.0000	0.00	99.96	99.96	0.007	0.088	0.088
st_qpc-m0	-6.0000	-5.0000	0.00	99.96	99.96	0.007	0.015	0.023
st_qpc-m1	-612.2714	-473.7778	0.00	99.99	99.98	0.009	0.223	0.233
st_qpc-m3a	-725.0518	-382.6950	0.00	98.10	99.16	0.025	3615.442	3727.123
st_qpc-m3b	-24.6757	0.0000	0.00	100.00	100.00	0.021	0.566	1.648
st_qpk1	-11.0000	-3.0000	0.00	99.98	99.98	0.007	0.110	0.053
st_qpk2	-21.0000	-12.2500	0.00	71.34	83.33	0.025	3599.788	3622.692
st_qpk3	-66.0000	-36.0000	0.00	33.53	50.04	0.077	3621.930	3778.200
st_rv1	-64.2359	-59.9439	0.00	96.19	98.44	0.023	3607.723	3602.339
st_rv2	-73.0007	-64.4807	0.00	88.79	81.85	0.079	3601.528	44.550
st_rv3	-38.5155	-35.7607	0.00	40.40	72.68	0.108	112.028	3807.828
st_rv7	-148.9816	-138.1875	0.00	45.43	62.28	0.269	3640.861	3880.783
st_rv8	-143.5829	-132.6616	0.00	29.90	45.80	0.663	3696.452	3874.801
st_rv9	-134.9131	-120.1164	0.00	20.56	31.64	1.019	3920.213	3675.654
st_z	-0.9674	0.0000	0.00	99.96	99.95	0.009	2.749	0.790

**Table 6.** GLOBALLib Instances with zero Duality Gap

Instance	RLT	Opt	Spectral Norm of $Y - xx^T$		
			V1	V2	V3
st_e17	0.0019	0.0019	0.000000	0.000000	0.000000
st_qpc-m3c	0.0000	0.0000	0.000000	0.000000	0.000000
st_qpc-m4	0.0000	0.0000	0.000000	0.000000	0.000000
ex2_1_2	-213.0000	-213.0000	0.000000	0.000000	0.000000
ex2_1_4	-11.0000	-11.0000	0.000000	0.000000	0.000000
st_e42	18.7842	18.7842	0.000000	0.000000	0.000000
st_fp2	-213.0000	-213.0000	0.000000	0.000000	0.000000
st_fp4	-11.0000	-11.0000	0.000000	0.000000	0.000000
st_bpk1	-13.0000	-13.0000	0.000000	0.000000	0.000000
st_bpk2	-13.0000	-13.0000	0.000000	0.000000	0.000000
st_gimp_fp1	10.0000	10.0000	0.000000	0.000000	0.000000
st_ph10	-10.5000	-10.5000	0.000000	0.000000	0.000000
st_bpv1	10.0000	10.0000	0.027262	0.000007	0.000007
st_gimp_ss2	3.0000	3.0000	0.043577	0.000021	0.000021
st_gimp_kk90	3.0000	3.0000	0.021689	0.000022	0.000022
st_e34	0.0156	0.0156	0.064299	0.000030	0.000029
st_e01	-6.6667	-6.6667	0.056653	0.000046	0.000046
st_fp3	-15.0000	-15.0000	0.293089	0.302328	0.000139
ex2_1_3	-15.0000	-15.0000	0.297487	0.000962	0.000150
st_gimp_fp3	-12.0000	-12.0000	0.000637	0.000235	0.000235
ex14_1_2	0.0000	0.0000	0.171873	0.171873	0.001654
ex14_1_5	0.0000	0.0000	0.146196	0.229286	0.103878
ex14_1_6	0.0000	0.0000	0.182808	0.208698	0.219895
st_robot	0.0000	0.0000	0.230963	0.227246	0.215491

**Table 7.** GLOBALLib Instances with Linear Complementarity Constraints

Instance	RLT	Opt	% Duality Gap Closed		Time (sec)	
			V2	V3	V2	V3
ex9_1_4	-63.0000	-37.0000	100.00	99.97	2.462	22.418
ex9_2_1	-16.0000	17.0000	99.95	99.95	3609.323	2351.308
ex9_2_2	-50.0000	100.0000	100.00	100.00	401.642	743.086
ex9_2_3	-30.0000	0.0000	99.99	99.99	27.718	522.123
ex9_2_4	-396.0000	0.5000	99.99	100.00	3.547	5.136
ex9_2_6	-406.0000	-1.0000	80.22	92.09	338.001	3652.873
ex9_2_7	-9.0000	17.0000	99.97	99.95	3607.258	3478.207

instance, when amended with disjunctive cuts from the linear complementarity constraints it closes 100% of the duality gap.

Next we present our computational results on the **MIQCP** instances proposed in [12]. These problem have both continuous and integer variables and quadratic constraints. They are of relatively small size with between 10 and 54 variables. Table 2 summarizes the experiment. RLT is the value of the RLT relaxation,

Table 8. Box QP Instances

Instance	wRLT	OPT	% Duality Gap Closed			Time (sec)		
			V1	V2	V3	V1	V2	V3
spar020-100-1	-1137	-706.5	58.66	95.40	99.64	3635.459	3638.200	3646.691
spar030-090-3	-2619.5	-1494	60.25	86.37	92.68	3730.348	3701.849	3607.885
spar040-060-2	-3011	-2004.23	43.05	55.79	61.63	3813.728	3707.992	3879.912
spar020-100-2	-1328.5	-856.5	70.36	93.08	97.81	3629.580	3636.665	3634.559
spar030-100-1	-2683.5	-1227.13	59.97	81.10	87.48	3647.126	3692.504	3624.834
spar040-060-3	-3532	-2454.5	56.60	72.63	79.30	3688.747	3764.079	3716.242
spar020-100-3	-1224	-772	70.70	97.47	99.97	3609.973	3632.560	3621.301
spar030-100-2	-2870.5	-1260.5	50.56	72.87	82.52	3662.868	3697.329	3753.816
spar040-070-1	-3194.5	-1605	53.82	64.03	70.28	3716.161	3642.681	3929.653
spar030-060-1	-1472.5	-706	32.55	60.00	73.32	3685.753	3823.051	3742.955
spar030-100-3	-2831.5	-1511.05	63.32	84.10	90.29	3712.164	3606.496	3682.094
spar040-070-2	-3446.5	-1867.5	45.84	57.91	63.86	3695.329	3756.377	3767.655
spar030-060-2	-1741	-1377.17	62.19	91.16	93.04	3731.242	3715.979	3748.334
spar040-030-1	-1162	-839.5	14.16	31.05	42.21	3694.667	3719.223	3874.422
spar040-070-3	-3833.5	-2436.5	50.57	62.94	69.89	3783.908	3693.666	3656.632
spar030-060-3	-2073.5	-1293.5	53.27	77.41	85.36	3666.710	3696.495	3702.028
spar040-030-2	-1695	-1429	13.92	27.74	31.29	3814.827	3937.898	3910.581
spar040-080-1	-3969	-1838.5	42.80	58.37	64.47	3710.865	3808.258	3811.056
spar030-070-1	-1647	-654	30.74	57.39	70.49	3685.224	3786.025	3679.571
spar040-030-3	-1322	-1086	2.35	28.00	34.74	3639.965	3798.683	4079.434
spar040-080-2	-3902.5	-1952.5	51.27	66.96	71.16	3667.295	4062.433	3845.179
spar030-070-2	-1989.5	-1313	61.19	86.60	92.26	3642.745	3708.212	3653.440
spar040-040-1	-1641	-837	17.42	33.31	37.70	3689.320	3817.844	3883.183
spar040-080-3	-4440	-2545.5	61.18	72.31	77.20	3703.711	4057.149	3806.478
spar030-070-3	-2367.5	-1657.4	73.58	88.66	92.85	3680.997	3744.044	3731.627
spar040-040-2	-1967.5	-1428	24.27	35.19	39.92	3839.449	3968.111	3667.330
spar040-090-1	-4490	-2135.5	54.63	66.64	72.50	3715.925	3781.044	3977.672
spar030-080-1	-2189	-952.729	41.71	69.67	78.41	3706.572	3600.777	3715.601
spar040-040-3	-2089	-1173.5	14.76	26.71	30.88	3718.280	3972.902	4002.336
spar040-090-2	-4474	-2113	55.86	66.46	70.59	3815.415	3931.349	3615.504
spar030-080-2	-2316	-1597	53.96	86.25	92.48	3690.453	3627.132	3702.961
spar040-050-1	-2204	-1154.5	23.12	36.72	43.34	3750.454	3819.720	3619.095
spar040-090-3	-4641	-2535	61.08	73.49	78.86	3808.143	4003.706	3777.561
spar030-080-3	-2504.5	-1809.78	69.28	91.42	95.70	3642.447	3666.392	3735.913
spar040-050-2	-2403.5	-1430.98	27.17	40.87	48.62	3738.085	3610.640	3757.075
spar040-100-1	-5118	-2476.38	65.26	76.24	79.10	3848.559	3853.573	3631.410
spar030-090-1	-2521	-1296.5	54.64	81.15	89.47	3702.696	3676.815	3657.596
spar040-050-3	-2715	-1653.63	20.75	33.95	43.11	3709.104	3639.977	3865.383
spar040-100-2	-5043	-2102.5	54.47	63.89	70.40	3759.668	3658.261	3771.344
spar030-090-2	-2755	-1466.84	56.33	82.66	88.79	3658.607	3646.756	3663.516
spar040-060-1	-2934	-1322.67	35.83	47.75	54.57	3648.720	3760.964	3724.381
spar040-100-3	-5196.5	-1866.07	52.41	59.92	65.08	3712.925	3842.685	3950.384

Opt is the value of the global optimum of the problem and V1, V2 and V3 give the strengthened bound obtained by each of the three variants. As can be seen from the results Variants 2 and 3 close almost all the gap for the second and third instance. For the first and fourth example, the gap closed is not as much, but in all cases variant 2 and 3 close substantially more gap than variant 1.

Next, we present our results on box constrained QPs. The test bed consists of a subset the test problems used in [26]. These problems are randomly generated box QPs with  $A_0$  of various densities. For this experiment we ran the three variants of our cut-generation procedures on the 42 problems with 20, 30 and 40 variables. We found that the RLT relaxation of these problem when amended with the convex quadratic cuts already closes around 95% of the duality gap. Hence, in order to better evaluate the performance of our cutting planes, we weakened the initial RLT relaxation (referred to as wRLT in the sequel) by removing the inequalities  $y_{ii} \leq x_i$ ; these inequalities are envelope inequalities associated with the product term  $y_{ii} = x_i x_i$ .

Table 8 summarizes the experiments. The second column of the table reports the optimal value of the wRLT relaxation, whereas the third column of the table gives the value of the optimal solution as reported in [26]. Overall, Variant 1 closes substantially less gap than variants 2 and 3. On average the amount of gap closed by Variant 1 is 46.81% while Variant 2 closes 65.28% and Variant 3 closes 71.51%.

## Acknowledgments

Part of this work was done when the first author was visiting the IBM T.J. Watson Research Center at Yorktown Heights, and their support is kindly acknowledged. Research of the first author was also supported by the National Science Foundation through grant DMI-0352885 and by the Office of Naval Research through contract N00014-03-1-0133. Research of the second author was carried out in part while affiliated with IBM T.J. Watson Research Center. Research of the second author was also supported by ANR grant BLAN06-1-138894. Thanks to Sam Burer for providing the box-QP instances.

## References

1. Anstreicher, K.M.: Semidefinite Programming versus the Reformulation-Linearization Technique for Nonconvex Quadratically Constrained Quadratic Programming. Preprint. Optimization Online (May 2007)
2. Balas, E.: Disjunctive programming: properties of the convex hull of feasible points. *Disc. Appl. Math.* 89(1-3), 3–44 (1998)
3. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Program.* 58, 295–324 (1993)
4. Balas, E., Saxena, A.: Optimizing over the split closure. MSRR# 674, Tepper School of Business, Carnegie Mellon Univ., *Math. Program. A* (to appear, 2005)
5. Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A.: An Algorithmic Framework for Convex Mixed-integer Nonlinear Programs. *Discrete Optimization* (in press)

6. Burer, S., Vandembussche, D.: A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Math. Programming* (to appear)
7. Cornuéjols, G.: Revival of the Gomory cuts in the 1990's. *Annals of Operations Research* 149(1), 63–66 (2007)
8. Fischetti, M., Lodi, A.: Optimizing over the first Chvatal closure. *Mathematical Programming* (to appear)
9. Fletcher, R., Leyffer, S.: User Manual for FilterSQP. Numerical Analysis Report NA/181, Dundee University (1998)
10. Jeroslow, R.G.: There cannot be any algorithm for integer programming with quadratic constraints. *Operations Research* 21(1), 221–224 (1973)
11. GLOBALLib, <http://www.gamsworld.org/global/globallib/globalstat.htm>
12. Lee, S., Grossmann, I.E.: A global optimization algorithm for nonconvex generalized disjunctive programming and applications to process systems. *Computers and Chemical Engineering* 25, 1675–1697 (2001)
13. Kim, S., Kojima, M.: Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optim. Methods and Software* 15, 201–204 (2001)
14. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I Convex underestimating problems. *Math. Prog.* 10, 147–175 (1976)
15. Nowak, I., Alperin, H., Vigerske, S.: LaGO - An object oriented library for solving MINLPs. In: Blicq, C., et al. (eds.) *Global Optimization and Constraint Satisfaction*, pp. 32–42. Springer, Berlin (2003)
16. Nowak, I.: *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*. Birkhauser, Basel (2005)
17. Vigerske, S.: <http://projects.coin-or.org/LaGO>
18. Saxena, A., Goyal, V., Lejeune, M.: MIP Reformulations of the Probabilistic Set Covering Problem (2007) *Optimization Online* (e-print), [http://www.optimization-online.org/DB\\_HTML//02/1579.html](http://www.optimization-online.org/DB_HTML//02/1579.html)
19. Sen, S.: Relaxations for probabilistically constrained programs with discrete random variables. *Operations Research Letters* 11(2), 81–86 (1992)
20. Sherali, H.D., Adams, W.P.: *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*. Kluwer, Dordrecht (1998)
21. Sherali, H.D., Fraticelli, B.M.P.: Enhancing RLT relaxations via a new class of semidefinite cuts. *J. Global Optim.* 22, 233–261 (2002)
22. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Methods and Software* 11-12, 625–653 (1999)
23. Tawarmalani, M., Sahinidis, N.V.: *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, Boston (2002)
24. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed integer nonlinear programs: A theoretical and computational study. *Math. Prog.* 99(3), 563–591 (2004)
25. Vandembussche, D., Nemhauser, G.L.: A polyhedral study of nonconvex quadratic programs with box constraints. *Math. Prog.* 102(3), 531–556 (2005)
26. Vandembussche, D., Nemhauser, G.L.: A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Math. Prog.* 102(3), 559–575 (2005)
27. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Prog.* 106(1), 25–57 (2006)

# The Air Traffic Flow Management Problem: An Integer Optimization Approach

Dimitris Bertsimas<sup>1</sup>, Guglielmo Lulli<sup>2</sup>, and Amedeo Odoni<sup>3</sup>

<sup>1</sup> Sloan School of Management and Operations Research Center, M.I.T.  
dbertsim@mit.edu

<sup>2</sup> Dept. of Informatics, Systems and Communication, University of Milano “Bicocca”  
guglielmo.lulli@disco.unimib.it

<sup>3</sup> Dept. of Aeronautics and Astronautics and Operations Research Center, M.I.T.  
arodoni@mit.edu

**Abstract.** In this paper, we present a new Integer Program (IP) for the Air Traffic Flow Management (ATFM) problem. The model we propose provides a complete representation of all the phases of each flights, i.e., the phase of taking-off, of cruising and of landing; suggesting all the actions to be implemented to achieve the goal of safe, efficient, and expeditious aircraft movement. The distinctive feature of the model is that it allows rerouting decisions. These decisions are formulated by means of “local” conditions, which allow us to represent such decisions in a very compact way by only introducing new constraints. Moreover, to strengthen the polyhedral structure of the underlying relaxation, we also present three classes of valid inequalities.

We report short computational times (less than 15 minutes) on instances of the size of the US air traffic control system that make it realistic that our approach can be used as the main engine of managing air traffic in the US.

## 1 Introduction

The continuous growth of the air transportation industry have put an enormous strain on the aviation system. Congestion phenomena are persistent and arise almost on a daily basis as a consequence of bad weather conditions which cause sudden capacity reductions. In the year 2000, approximately one out of every four flights in the United States was delayed or canceled, [6]. The resulting delays have a significant economic impact. The Air Transport Association has estimated that system delays drove an estimated \$5.9 billion in direct operating costs for United States airlines in 2005. Similar figures have been shown by European airlines.

As a result, air traffic flow management (ATFM) has become increasingly crucial. ATFM attempts to prevent local demand-capacity imbalances by adjusting the flows of aircraft on a national or regional basis. Until now, the ATFM have been mainly focusing on airports’ congestion. On this subject, the most popular approach, by far, has been the allocation of ground delays to departing flights, i.e., postponing their departure time. From the paper by Odoni [9], who was the first to formalize this problem, a plethora of models and algorithms have been developed to detect optimal strategies to assign ground delays to flights (see [1] and [6]).



However, it has become increasingly evident that very significant delays and system throughput degradations have arisen from en-route airspace problems and limitations. The problem posed by the en-route sector capacity constraints is persistent and may take at least one more decade to resolve [4]. One of the implications of the simultaneous presence of airport and en-route airspace constraints is that devising good strategies is a much more complicated task. Any mathematical model developed for this purpose has to consider a true network of capacitated elements, en-route sectors and airports [8]. Moreover, a larger set of options to resolve congestion is available: ground holding, airborne holding, miles-in-tails and rerouting, i.e., the possibility of reroute a flight on a different flight path if the current route passes through a region that unexpectedly becomes congested.

As opposed to the airport congestion case, the research literature dealing with en-route congestion is quite sparse. One of the first attempts to include in the ATFM problem en-route capacity restrictions was by Helme [5], who proposed a multi-commodity minimum-cost flow on a time-space network to assign airborne and ground delay to aggregate flow of flights, commodities of the network flow model. While the formulation of this model is straightforward and easy to understand, its computational performance was rather weak. Lindsay et al. [7] formulated a disaggregate deterministic 0-1 integer programming models for deciding ground and airborne holding of individual flights in presence of both airport and airspace capacity constraints. Bertsimas and Stock [2] presented a deterministic 0-1 IP model to solve a similar problem. The model decides on the departure time and sector occupancy time of each aircraft. The model enables very efficient computation of optimal solutions, since several of the constraints provide facets of the convex hull of solutions. However this model, as well as those cited above, does not consider rerouting as an option. It assumes that the flight path is known in advance and is fixed.

To the best of our knowledge the only work which considers rerouting, at a least at a macroscopic level, is the work by Bertsimas and Stock Patterson [3]. They presented a dynamic, multi-commodity, integer network-flow model. The model addressed routing as well as scheduling decisions, but it did not provide computational performances aligned with the dimensions of real instances.

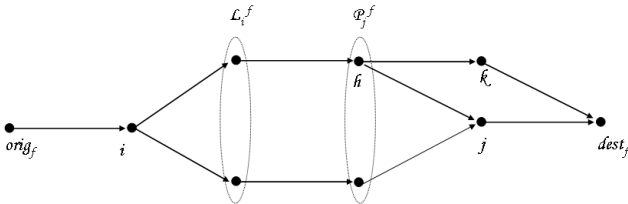
Modeling rerouting decisions has posed one of the greatest challenges in this field of research. Our goal is to combine the model “flexibility” in terms of range of decisions of model presented in [3] with the shown mathematical properties of the model presented in [2], so that we are able to solve efficiently sized problems. Herein, we present a mathematical model for the ATFM which includes all the possible options to resolve air congestion, including rerouting. The scope of the model is to suggest the time of departure, the route, the time required to cross each sector and the time of arrivals taking into account the capacity of all sectors and airports. The main feature of the model is the formulation of rerouting decisions in a very compact way. With respect to previous models, the methodology we presented does not require any additional variables, but it only introduces new constraints. These constraints implement local routing conditions that are sufficient for the purpose of the model. To strengthen the polyhedral structure of the underlying relaxation, we also present three classes of valid inequalities.

The paper is organized as follows: In Section 2, we present the mathematical model for the ATFM problem with rerouting, and three classes of valid inequalities as well. The computational experience is reported in Section 3. Finally, Section 4 contains conclusions and indications for future research.

## 2 The Mathematical Model

The mathematical model we present here, is intended to determine how to adjust the release time of each flight into the system (time of departure), how to control its flight speed once in the air and how to reroute it in case of sectors' congestion along the preferred path. As an underlying model we consider the model proposed by Bertsimas and Stock [2].

Any origin-destination route is represented as a sequence of sectors flown by an aircraft. In ATFM models which do not include rerouting as an option, the sequence of sectors to be flown is pre-determined. To contemplate rerouting in the mathematical model the set of possible sectors that might be flown has to be enlarged.



**Fig. 1.** Given a flight  $f$ , the set  $\mathcal{L}_i^f$  of sectors that follow sector  $i$ , and the set of sectors  $\mathcal{P}_i^f$  that precede sector  $j$

A key element of the proposed model is the definition of routes. The origin-destination routes can be represented by digraphs. The set of nodes of the digraph ( $\mathcal{S}_f$ ) represents the set of capacitated elements of the airspace, e.g., airports and sectors. The set of arcs defines the sequence relations. There is an arc from a node  $v$  to node  $w$  if  $v$  and  $w$  are contiguous sectors and sector  $w$  can be flown soon after sector  $v$ . In Figure 1, three different routes between the airport of origin and of destination are reported. Within the ATFM framework, we may suppose, without loss of generality, that the digraph of  $f$  routes is acyclic. This allows us to equip the set of sectors with a binary relation, and hence envision the set of possible routes within the framework of so-called partially ordered set (poset). The airport of departure and arrival are the minimum and the maximum elements of the poset respectively. The set of possible routes between the  $v, w$  pair corresponds to the set of maximal chains of the poset. To impose that each flight follows exactly one route we use local conditions, that can be simply stated as follows:

- to fly a sector any aircraft has first to fly one of the previous sectors for at least a number of time periods equal to their sector flight time;

or equivalently,

- if an aircraft is flying a sector, for at least a number of time periods equal to its flight time, then immediately after it will fly only one of the subsequent sectors.

To formally describe these routing conditions we introduce the following additional notation. For each sector  $\nu$  ( $\in \mathcal{S}_f$ ) the subset of sectors which follow  $\nu$  is denoted by  $\mathcal{L}_i^f \subset \mathcal{S}_f$ . Analogously the subset of sectors that precede  $\nu$  is denoted by  $\mathcal{P}_i^f \subset \mathcal{S}_f$  (see Figure III).

In what follows, we call  $\nu_1, \nu_2, \dots$  all the sectors followed by more than one sector, e.g., Sector  $\nu$  and Sector  $\nu'$  in Figure III, while those sectors preceded by more than one sector are called  $\nu_1, \nu_2, \dots$ , Sector  $\nu'$  in the same figure.

## 2.1 The Mathematical Formulation

The model's formulation requires definition of the following notation:

- $\mathcal{K} \equiv$  set of airports,
- $\mathcal{S} \equiv$  set of sectors,
- $\mathcal{S}^f \subseteq \mathcal{S} \equiv$  set of sectors that can be flown by flight  $f$ ,
- $\mathcal{F} \equiv$  set of flights,
- $\mathcal{T} \equiv$  set of time periods,
- $\mathcal{C} \equiv$  set of pairs of flights that are continued,
- $\mathcal{P}_i^f \equiv$  set of sector  $\nu$ 's subsequent sectors,
- $\mathcal{L}_i^f \equiv$  set of sector  $\nu$ 's previous sectors,
- $D_k(t) \equiv$  departure capacity of airport  $k$  at time  $t$ ,
- $A_k(t) \equiv$  arrival capacity of airport  $k$  at time  $t$ ,
- $S_j(t) \equiv$  capacity of sector  $j$  at time  $t$ ,
- $d_f \equiv$  scheduled departure time of flight  $f$ ,
- $a_f \equiv$  scheduled arrival time of flight  $f$ ,
- $s_f \equiv$  turnaround time of an airplane after flight  $f$ ,
- $orig_f \equiv$  airport of departure of flight  $f$ ,
- $dest_f \equiv$  airport of arrival of flight  $f$ ,
- $l_{fj} \equiv$  number of time units that flight  $f$  must spend in sector  $j$ ,
- $T_j^f = [\underline{T}_j^f, \bar{T}_j^f] \equiv$  set of feasible time periods for flight  $f$  to arrive in sector  $j$ ,
- $\underline{T}_j^f \equiv$  first time period in the set  $T_j^f$ ,
- $\bar{T}_j^f \equiv$  last time period in the set  $T_j^f$ .

**The Decision Variables.** As mentioned above, the model herein presented is based on the Bertsimas-Stock model [2] and we use the same decision variables.

$$w_{j,t}^f = \begin{cases} 1, & \text{if flight } f \text{ arrives at sector } j \text{ by time } t, \\ 0, & \text{otherwise.} \end{cases}$$

This definition of the decision variables ( $w_{j,t}^f$ ) using “.” instead of “ $\cdot$ ” is critical to the understanding of the formulation. If flight  $f$  arrives at time  $t$  at sector  $j$  then both variable of time period  $t$  and subsequent ones will be set to 1 (i.e.,  $w_{j,\tau}^f = 1 \ \forall \tau \geq t$ ).

**The Objective Function.** As in most other ATFM models in the literature, the model we propose minimizes a cost function which is a combination of both airborne-holding delay (AH) and ground holding delay (GH), of the form  $\alpha \cdot AH + GH$  with  $\alpha > 1$ . For convenience, we re-write the objective function as  $\alpha \cdot TD - (\alpha - 1) \cdot GH$ , being  $TD (= AH + GH)$  the total delay.

To ensure equity among flights, we include in the objective function cost coefficients that are a super-linear function of the tardiness of a flight of the form  $(t - a_f)^{1+\epsilon}$ , with  $\epsilon$  close to zero. This will favour the assignment of a moderate amount of total delay to each of two flights rather than the assignment of a small amount to one and a large amount to the other.

For each flight  $f$  and for each time period  $t$ , we define the following cost coefficients:

$$\begin{aligned} c_{td}^f(t) &= (t - a_f)^{1+\epsilon} \equiv \text{total cost of delaying flight } f \text{ for } (t - a_f) \text{ unit of time,} \\ c_g^f(t) &= (\alpha - 1)(t - d_f)^{1+\epsilon} \equiv \text{cost reduction for holding flight } f \text{ on the ground} \\ &\quad \text{for } (t - d_f) \text{ unit of time.} \end{aligned}$$

In view of the description above, the objective function is as follows:

$$\text{Min} \sum_{f \in \mathcal{F}} \left( \sum_{t \in T_{dest_f}^f} c_{td}^f(t) \cdot (w_{dest_f,t}^f - w_{dest_f,t-1}^f) - \sum_{t \in T_{orig_f}^f} c_g^f(t) \cdot (w_{orig_f,t}^f - w_{orig_f,t-1}^f) \right)$$

## The Constraints

$$\sum_{f \in \mathcal{F}: orig_f = k} (w_{k,t}^f - w_{k,t-1}^f) \leq D_k(t) \quad \forall k \in \mathcal{K}, \ t \in \mathcal{T}. \quad (1)$$

$$\sum_{f \in \mathcal{F}: dest_f = k} (w_{k,t}^f - w_{k,t-1}^f) \leq A_k(t) \quad \forall k \in \mathcal{K}, \ t \in \mathcal{T}. \quad (2)$$

$$\sum_{f \in \mathcal{F}: j \in \mathcal{S}_f} (w_{j,t}^f - \sum_{j' \in \mathcal{L}_i^f} w_{j',t}^f) \leq S_j(t) \quad \forall j \in \mathcal{S}, \ t \in \mathcal{T}. \quad (3)$$

$$w_{j,t}^f \leq \sum_{j' \in \mathcal{P}_j^f} w_{j',t-l_{fj'}}^f \quad \forall f \in \mathcal{F}, \ t \in T_j^f, \ j \in \mathcal{S}^f : j \neq orig_f. \quad (4)$$

$$w_{j,\bar{T}_j^f}^f \leq \sum_{j' \in \mathcal{L}_j^f} w_{j',\bar{T}_{j'}^f} \quad \forall f \in \mathcal{F}, j \in \mathcal{S}^f : j \neq dest_f. \quad (5)$$

$$\sum_{j' \in \mathcal{L}_j^f} w_{j',\bar{T}_{j'}^f} \leq 1 \quad \forall f \in \mathcal{F}, j \in \mathcal{S}^f : j \neq dest_f. \quad (6)$$

$$w_{orig_f,t}^f - w_{dest_{f'},t-s_f}^{f'} \leq 0 \quad \forall (f, f') \in \mathcal{C}, \forall t \in T_k^f. \quad (7)$$

$$w_{j,t-1}^f - w_{j,t}^f \leq 0 \quad \forall f \in \mathcal{F}, j \in \mathcal{S}^f, t \in T_j^f. \quad (8)$$

$$w_{j,t}^f \in \{0, 1\} \quad \forall f \in \mathcal{F}, j \in \mathcal{S}^f, t \in T_j^f. \quad (9)$$

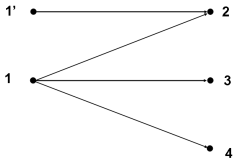
The first three sets of constraints take into account the capacities of various aspects of the system. Constraints (1) ensure that the number of flights which may take off from airport  $\bar{a}$  at time  $\tau$ , will not exceed the departure capacity of airport  $\bar{a}$  at time  $\tau$ . Likewise, Constraints (2) ensure that the number of flights which may arrive at airport  $\bar{a}$  at time  $\tau$ , will not exceed the arrival capacity of airport  $\bar{a}$  at time  $\tau$ . Finally, Constraints (3) ensure that the sum of all flights which may feasibly be in Sector  $\bar{c}$  at time  $\tau$  will not exceed the capacity of Sector  $\bar{c}$  at time  $\tau$ . This difference gives the flights that are in Sector  $\bar{c}$  at time  $\tau$ , since the first term will be 1 if Flight  $\bar{a}$  has arrived in sector  $\bar{c}$  by time  $\tau$  and the second term will be 1 if flight  $\bar{a}$  has arrived at one of the next sectors by time  $\tau$ . So, the only flights that will contribute a value of 1 to this sum are those flights that have arrived at  $\bar{c}$  and have not yet departed from  $\bar{c}$  by time  $\tau$ . Constraints (4), (5) and (6) represent connectivity between sectors. They stipulate that a flight can not arrive at Sector  $\bar{c}$  by time  $\tau$  if it has not arrived to one of the previous sectors by time  $t - l_{fj'}$ . In other words, a flight cannot enter the next sector on its path until it has spent  $l_{fj'}$  time units (the minimum possible) traveling through one of the previous sectors in its current path. Moreover, Constraints (5) and (6) state that a flight will certainly arrive to one of the subsequent sectors. Constraints (7) represent connectivity between airports. They handle the cases in which a flight is continued, i.e., the flight's aircraft is scheduled to perform a later flight within some time interval. We will call the first flight  $f'$  and the following flight  $f$ . Constraints (8) represent connectivity in time. Thus, if a flight has arrived by time  $\tilde{t}$ , then  $w_{j,t}^f$  has to have a value of 1 for all later time periods ( $t \geq \tilde{t}$ ).

In what follows, we present three classes of valid inequalities with the scope of strengthening the formulation.

**Proposition 1:**  $w_{j,t}^f = 0 \implies w_{j',t+l_{fj'}}^f = 0 \quad \forall f \in \mathcal{F}, j \in \mathcal{S}^f, t \in T_j^f, j' \in \mathcal{P}_j^f, |\mathcal{P}_j^f| = 1$

$$w_{j,t}^f \geq \sum_{j' \in \mathcal{L}_j^f : |\mathcal{P}_{j'}^f| = 1} w_{j',t+l_{fj'}}^f \quad \forall f \in \mathcal{F}, t \in T_j^f.$$

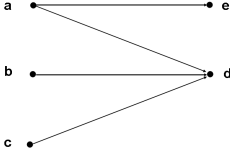
The inequalities of Proposition 1 state that if a flight  $\bar{a}$  has not crossed Sector  $\bar{c}$  by time  $\tau$  ( $w_{j,t}^f = 0$ ), it will not cross any of the subsequent sectors by time  $t + l_{fj'}$  unless these subsequent sectors can be reached from elsewhere, as in the



$$\text{cons. (4)} \begin{cases} w_{2,t} \leq w_{1,t-l} + w_{1',t-l} \\ w_{3,t} \leq w_{1,t-l} \\ w_{4,t} \leq w_{1,t-l} \end{cases}$$

$$\text{v.i.1 } w_{3,t} + w_{4,t} \leq w_{1,t-l}$$

Fig. 2. Valid inequality for a fork node



$$\text{cons. (5)} \begin{cases} w_{a,T} \leq w_{d,T} + w_{e,T} \\ w_{b,T} \leq w_{d,T} \\ w_{c,T} \leq w_{d,T} \end{cases}$$

$$\text{v.i.2 } w_{b,T} + w_{c,T} \leq w_{d,T}$$

Fig. 3. Valid inequality for a joint sector

case of  $w_{j,t}^f = 0$  in Figure 2. The fork inequalities hold by Constraints (5) and (8) in case  $w_{j,t}^f = 1$  and by Constraints (4) in case  $w_{j,t}^f = 0$ .

Conditions given above can be extended to the case of a joint sector. If a flight  $f$  do not cross Sector  $j$  then it have not crossed any of the previous sectors unless these sectors are also adjacent to other sectors. Hence, let us restrict the attention to sectors which are only adjacent to the joint sector among all the previous ones, e.g., sectors  $a$  and  $b$  of the example in Figure 3.

**Proposition 2:** Let  $\mathcal{F}$  be a set of flights and  $\mathcal{P}_j^f$  be the set of previous sectors adjacent to sector  $j$  for flight  $f$ .

$$\sum_{j' \in \mathcal{P}_j^f : |\mathcal{L}_{j'}^f| = 1} w_{j',T_{j'}^f} \leq w_{j,T_j^f} \quad \forall f \in \mathcal{F}.$$

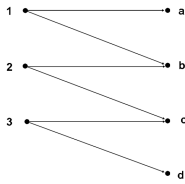
The joint inequalities hold by Constraints (5) if  $w_{j,t}^f = 0$  and by Constraints (4) if  $w_{j,t}^f = 1$ .

The network of possible routes, represented by an acyclic graph, naturally defines a preorder relation on the set of sectors. Each  $(j, j')$  pair corresponds to a chain of the poset and the Proposition in the sequel immediately follows:

**Proposition 3:** Let  $\mathcal{A}$  be a set of sectors and  $\mathcal{S}^f$  be the set of sectors crossed by flight  $f$ .

$$\sum_{j \in \mathcal{A}} w_{j,T_j^f} \leq 1.$$

These conditions state that each flight follows exactly one route.



$$\text{cons. (6)} \begin{cases} w_{a,T} + w_{b,T} \leq 1 \\ w_{b,T} + w_{c,T} \leq 1 \\ w_{c,T} + w_{d,T} \leq 1 \end{cases}$$

a.i.  $w_{a,T} + w_{b,T} + w_{c,T} + w_{d,T} \leq 1$

Fig. 4. Antichain inequality

**Proposition 4:**

$$\{j \in \mathcal{L}_i^f : |\mathcal{P}_j^f| = 1\} = \mathcal{L}_i^f \quad \square$$

### 3 Computational Experience

In this section, we present the computational experience on the mathematical model presented in §2.1, including the valid inequalities given in Proposition 1 - Proposition 4. We consider randomly generated instances whose dimension is comparable to realistic ones. In particular, we consider two sets of instances. The first one represents the ATFM problem at a regional level, e.g., east-cost or mid-west US, while the second set, of larger instances, is more representative of the Nation wide problem. The size of the instance depends on the time horizon, the time discretization period, the number of sectors and airports and the demand at each airport. By changing one or all the parameters above, we generate different size instances.

The airspace is divided into equal size sectors, forming a grid. We also suppose that the minimum amount of time to fly a sector is the same for all the flights and for all the sectors. In order to generate instances which are consistent with the hub-and-spoke operations we cluster airports into hubs and regional airports. There are no flights connecting two regional airports, i.e., regional airports do not have direct connections but they are connected to hubs. For each airport, the demand of flights is randomly generated, drawn from a uniform distribution. Again with the sake of being more adherent to real operations, we generate hubs' demands considering both peak and off-peak periods. The average value of the demand for peak (off-peak) periods is set equal to 15 (8) flights per period. The nominal capacity of sectors and airports (capacity under good weather conditions) is set to values which allow to accommodate all the air traffic without incurring in too much air congestion.

To enforce sectors congestion, we suppose a capacity reduction of some sectors. The reduction of capacity affects 3 sectors at a time, for 5 consecutive time periods. Afterwards, three contiguous sectors experience capacity reduction for other 5 consecutive time periods, and so on. In this way, we “simulate” the effect of a bad weather front which move along a certain direction. We also consider instances with larger weather front and with different speed (number of time periods it stays in the sectors before moving forward), without affecting

the computational performance of the model. Herein we do not report all the computational results for the sake of brevity.

One of the key elements of our model is the set  $\mathcal{S}^f$  of sectors that can be flown by flight  $f$ . By default, all the sector on the shortest,  $l$ -route of flight  $f$  are included in  $\mathcal{S}^f$ . If one or more of the sectors on the shortest route is congested, additional sectors, those contiguous, are included in the set  $\mathcal{S}^f$ . For this purpose, sectors are considered congested if their demand exceeds 80% of the capacity. The number of forks in a  $l$ -pair gives a lower estimate of the number of possible routes between the origin and the destination. On average, the number of forks is about 3, meaning that on average we have at least 3 routes between each  $l$ -pair, even though it can be much larger. In a case with 20 forks, we counted 121  $l$ -routes.

### 3.1 Regional Size Instances

The computational results for “regional” instances are reported here. These instances include 20 airports, 10 of which are hubs, and 113 sectors which correspond to about one third of the NAS airspace. We consider a five-hour time horizon subdivided into 20 15-minute time units. All the instances manage roughly 3000 flights. The nominal capacity (capacity under good weather conditions) of sectors is set to 71 flights per period. Five sets of instances have been considered, each with a different percentage of flight connections, as reported in the first column of Table 1 (% of Conn.s). For instance, 50 indicates that half of the flights have a flight connection. In the first column, it is also reported between parenthesis the number of flights considered in the instance. Several scenarios for capacity reduction are tested, from the nominal value to values close to zero (sector closed), reported in the second column of the Table 1. These two parameters, i.e., percentage of connections and capacity, univocally identify each instance.

To compute optimal solutions we use the CPLEX-MIP solver 9.0, implemented using AMPL as modeling language on a PC AMD-Xeon 4 processors 3 GHz, 8 GB RAM with Linux Ubuntu 4.03 OS. With these input data, the mathematical program has the order of 270,000 constraints and 150,000 variables, after pre-processing. In the pre-processing phase about 160,000 constraints and 200,000 variables are eliminated. Given the size of the instances, we accept good solutions within an optimality gap of 1%. The gap of the solution is listed in the fourth column of Table 1. To solve these instances, we also took advantage of the CPLEX’s capabilities of generating constraints (cuts) based on polyhedral considerations. In particular, we enable moderate generation of clique cuts, setting the corresponding parameter to 1. The number of additional cuts of clique, implied bound (Bound) and Gomory type are listed in the fifth, sixth and seventh column of Table 1 respectively. Finally, in the last column, we report the value of the objective function with the intention to provide a clue on the amount of delay assigned.

What immediately appears from the computational results is that CPLEX can compute good solution, if not optimal, in all the cases. The average solution time is 241 seconds. In only one case, the instance with 50% of connections and



**Table 1.** Computational results

% of Conn.s (# of Flights)	Capacity	Solution					Iter.s	O.F. value	
		Time (secs.)	GAP (%)	CUTS					
				Clique	Bound	Gomory			
50 (3003)	3	Infeasible							
	4	167.0	0.66	1551	394	46	143528	2972.3	
	5	352.2	0.28	2483	486	47	143763	2677.2	
	10	225.1	0.45	3340	537	50	142399	1853.8	
	20	383.4	0.21	3700	510	57	135670	949.5	
	30	724.2	0.49	2046	513	68	130214	409.8	
	40	382.0	0.28	852	292	50	124178	164.3	
	50	116.1	0.00	1133	297	34	120932	129.0	
	60	161.3	0.00	1231	358	36	120613	123.9	
70	213.5	0.00	1276	360	37	120417	123.1		
60 (3027)	6	Infeasible							
	7	194.5	0.61	1050	311	37	144194	2883.3	
	10	215.7	0.63	2014	406	45	142092	2381.4	
	20	207.3	0.72	2420	358	43	134586	1471.9	
	30	480.9	0.02	2308	476	38	133793	829.5	
	40	390.8	0.00	2381	528	54	128156	455.4	
	50	191.3	0.00	1936	448	61	123506	342.5	
	60	213.4	0.41	2354	450	59	121359	297.6	
70	204.4	0.40	2404	452	57	120385	280.1		
70 (3140)	10	Infeasible							
	11	194.8	0.06	-	-	6	152028	3554.1	
	12	139.0	0.17	-	-	-	147699	3357.3	
	15	106.4	0.08	-	-	-	141554	2857.6	
	20	125.8	0.02	-	-	6	142124	2303.3	
	30	76.6	0.00	-	-	-	136489	1487.1	
	40	177.3	0.57	-	-	11	135555	949.8	
	50	109.4	0.20	-	-	10	127251	688.2	
	60	42.7	0.00	-	-	-	124369	612.8	
70	42.8	0.00	-	-	-	123612	567.0		
80 (3240)	11	Infeasible							
	12	196.4	0.16	1284	331	48	157695	3080.7	
	15	248.3	0.04	1341	320	39	153471	2638.3	
	20	271.6	0.00	2169	409	48	153777	2149.6	
	30	368.7	0.00	2039	393	45	146618	1398.6	
	40	326.5	0.47	1891	486	51	144897	930.0	
	50	203.9	0.00	1148	443	46	139034	660.2	
	60	237.6	0.98	1458	318	32	136602	564.7	
70	148.9	0.72	1130	296	32	133049	517.5		
90 (3196)	14	Infeasible							
	15	235.7	0.99	2320	421	49	155881	2947.0	
	20	303.7	0.02	2694	523	43	150234	2325.9	
	30	266.5	0.56	2374	440	55	148645	1581.0	
	40	327.6	0.75	1500	496	55	143615	1105.8	
	50	380.4	0.40	2461	436	62	138366	862.8	
	60	220.6	0.00	2512	395	51	134305	745.9	
70	307.7	0.04	982	351	52	134156	695.2		

30 flights per period of capacity, the solution time is larger than 10 minutes. This is also the only instance for which the algorithm branched (68 nodes) in order to compute the solution within the optimality tolerance. On average instances with 50% of flight connections require longer computational time. This trend is explained by the larger number of symmetries that this set of instances may have, i.e., between flights flying the same  $i, j$  pair at the same time. On the other side, the set of instances that shows better computational performances is the set with 70% of flight connections. This set exhibits by far the smallest computational time.

Moreover, the computational performances of the mathematical model do not degrade as we consider instances close to the “infeasibility border”, as experienced in [10] for instance.

### 3.2 National Size Instances

The instances of “national” size consider 30 airports, 10 of which are hubs, 145 sectors and 22 time periods. All the instances manage 6475 flights with 5180 connections (80%). For these instances the nominal capacity of the sector is set to 130 flights per period. The capacity of sectors affected by the bad weather front is here reported in percentage of the nominal capacity (first column of Table 2).

To solve these instances we use the same setting for the CPLEX parameters as in the regional case. In addition, a time limit of 3,600 seconds is imposed. With these input data, the mathematical program has the order of 570,000 constraints and 305,000 variables, after pre-processing. In the pre-processing phase about 280,000 constraints and 340,000 variables are eliminated (fixed).

For this set of instances the average computational time to provide a solution within 1% is 987 secs. The median value is much smaller, equal to 710 secs. Indeed, in one case, that is the instance with the effective capacity equal to

**Table 2.** Nation wide instances

Capacity (%)	O.F. value	Solution Time (secs.)	CUTS			Iter.s	GAP (%)
			Clique	Bound	Gomory		
0	Infeasible						
10	2620	891	14879	3664	55	266454	0.09
20	1672	643	11424	2917	32	256845	0.00
30	1108	988	7739	2356	60	244400	0.75
40	693	836	7920	1951	63	238985	0.00
50*	414	3600	5458	1743	67	242378	1.05
60	265	631	6232	1487	53	231241	0.00
70	154	889	5585	1151	77	227936	0.99
80	74	694	3632	1178	74	226179	0.00
90	23	710	5355	1009	69	221224	0.00
100	12	437	1346	707	65	218378	0.00

\* 332 Branch-and-Bound nodes have been generated

50% of its nominal value, the algorithm can't compute a good solution with 1% optimality gap within the time limit of 3600 secs. This is also the only case in which the algorithm requires the branching phase, exploring 332 branch-and-bound nodes during its execution. However, accepting a larger optimality tolerance, say 3%, then the algorithm computes a good solution in 569 secs. The other statistics for this solution are as follows: objective function value of the solution is 417 with an optimality gap of 2.58. During the pre-processing phase 5458 cuts of clique type, 1693 of implied bound-type and 67 of Gomory-type are added. For all the instances the statistics are listed in Table 2.

To evaluate the effect of rerouting, we compare the solutions of the ATFM model with and without rerouting respectively. When sectors' capacity is close to the nominal value the difference between the two solutions is rather small, both in terms of ground and airborne holding delay. But, as the sectors' capacity decreases, the benefits of rerouting increase, in terms of smaller amounts of both ground and airborne holding delay assigned. In the most congested case, capacity equal to 10% of its nominal value, the reduction of ground delay is almost 30% (from 949 to 733 time units). The airborne holding delays decrease as well, dropping from 150 time units to 126 with an improvement of 19%. It is important to note that such a benefit is gained with a small amount of rerouting actions. Even in the most congested case, the number of rerouted flights (220) is rather small, which corresponds to 3.4% of the total flights.

## 4 Conclusions

In this paper, we presented a new mathematical model for the Air Traffic Flow Management problem. The key feature of the model is that it also includes rerouting decisions and they are formulated in a very compact way. In fact, it does not require any additional variable, but it only introduces new constraints, which implements local routing conditions. We also presented three classes of valid inequalities with the scope of strengthening the polyhedral structure of the underlying relaxation.

A wide computational analysis on realistic instances demonstrated the viability of the proposed model. We solved realistic instances of the problem in short computational times, which are consistent with the decision process inside the ATFM Central Unit. Given that our approach includes all the air traffic control decisions (ground holding, air holding, adjusting speed of aircraft and rerouting) combined with the attractive computational times, makes us optimistic that this approach may succeed in becoming the main air traffic control engine.

## References

1. Bertsimas, D., Odoni, A.: A critical survey of optimization models for tactical and strategic aspects of air traffic flow management. Technical report, NASA (1997)
2. Bertsimas, D., Stock, S.: The Air Traffic Management Problem with Enroute Capacities. *Operations Research* 46, 406–422 (1998)

3. Bertsimas, D., Stock Patterson, S.: The Traffic Flow Management Rerouting Problem in Air Traffic Control: A Dynamic Network Flow Approach. *Transportation Science* 34, 239–255 (2000)
4. EUROCONTROL Performance Review Commission, Performance Review Report, Brussels (2004)
5. Helme, M.: Reducing air traffic delay in a space-time network. In: *IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 236–242 (1992)
6. Hoffman, R., Mukherjee, A., Vossen, T.: Air Traffic Flow Management. Working Paper (2007)
7. Lindsay, K., Boyd, E., Burlingame, R.: Traffic flow management modeling with the time assignment model. *Air Traffic Control Quarterly* 1, 255–276 (1993)
8. Lulli, G., Odoni, A.R.: The European Air Traffic Flow Management Problem. *Transportation Science* 41, 1–13 (2007)
9. Odoni, A.R.: The Flow Management Problem in Air Traffic Control. In: Odoni, A.R., Bianco, L., Szego, G. (eds.) *Flow Control of Congested Networks*, pp. 269–288. Springer, Berlin (1987)
10. Vranas, P.B., Bertsimas, D., Odoni, A.R.: The Multi-Airport Ground Holding Problem in Air Traffic Control. *Operations Research* 42, 249–261 (1994)

# The Induced Disjoint Paths Problem

Ken-ichi Kawarabayashi<sup>1,\*</sup> and Yusuke Kobayashi<sup>2,\*\*</sup>

<sup>1</sup> National Institute of Informatics, Tokyo 101-8430, Japan

[k\\_keniti@nii.ac.jp](mailto:k_keniti@nii.ac.jp)

<sup>2</sup> University of Tokyo, Tokyo 113-8656, Japan

[Yusuke\\_Kobayashi@mist.i.u-tokyo.ac.jp](mailto:Yusuke_Kobayashi@mist.i.u-tokyo.ac.jp)

**Abstract.** For a graph  $G$  and a collection of vertex pairs  $\{(s_1, t_1), \dots, (s_k, t_k)\}$ , the disjoint paths problem is to find  $k$  vertex-disjoint paths  $P_1, \dots, P_k$ , where  $P_i$  is a path from  $s_i$  to  $t_i$  for each  $i = 1, \dots, k$ . This problem is one of the classic problems in combinatorial optimization and algorithmic graph theory, and has many applications, for example in transportation networks, VLSI layout, and recently, virtual circuits routing in high-speed networks.

As an extension of the disjoint paths problem, we introduce a new problem which we call the induced disjoint paths problem. In this problem we are given a graph  $G$  and a collection of vertex pairs  $\{(s_1, t_1), \dots, (s_k, t_k)\}$ . The objective is to find  $k$  paths  $P_1, \dots, P_k$  such that  $P_i$  is a path from  $s_i$  to  $t_i$  and  $P_i$  and  $P_j$  have neither common vertices nor adjacent vertices for any distinct  $i, j$ .

This problem setting is a generalization of the disjoint paths problem, since if we subdivide each edge, then desired disjoint paths would be induced disjoint paths. The problem is motivated by not only the disjoint paths problem but also the recognition of an induced subgraph. The latter has been developed in the recent years, and this is actually connected to the Strong Perfect Graph Theorem [4], and the recognition of the perfect graphs [2].

In this paper, we shall investigate the complexity issues of this problem. The induced disjoint paths problem has several variants depending on whether  $k$  is a fixed constant or a part of the input, whether the graph is directed or undirected, and whether the graph is planar or not. We show that the induced disjoint paths problem is

- (i) solvable in polynomial time when  $k$  is fixed and  $G$  is a directed planar graph,
- (ii) solvable in linear time when  $k$  is fixed and  $G$  is an undirected planar graph,
- (iii) NP-hard when  $k = 2$  and  $G$  is an acyclic directed graph,
- (iv) NP-hard when  $k = 2$  and  $G$  is an undirected general graph.

(i) generalizes the result by Schrijver [22], while (ii) generalizes the result by Reed, Robertson, Schrijver and Seymour [17].

---

\* Research partially supported by JSPS Postdoctoral Fellowships for Research Abroad.

\*\* Supported by the Research Fellowship of the Japan Society for the Promotion of Science for Young Scientists.

# 1 Introduction

## 1.1 Disjoint Paths Problem

The disjoint paths problem (DPP) is the following.

### Disjoint paths problem (DPP)

**Input:** A graph  $G$ ,  $k$  pairs of vertices  $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$  in  $G$  (which are sometimes called “terminals”).

**Output:** Pairwise vertex-disjoint paths  $P_1, \dots, P_k$  in  $G$  such that  $P_i$  joins  $s_i$  and  $t_i$  for  $i = 1, 2, \dots, k$ .

This problem is one of the classic problems in algorithmic graph theory and combinatorial optimization.

If  $k$  is as a part of the input of the problem, then this is one of Karp’s NP-complete problems [11], and it remains NP-complete even if  $G$  is constrained to be planar (Lynch [14]). The disjoint paths problem has been attracted interest from 1980’s. This is because it is closely related to transportation networks, VLSI layout, and recently, virtual circuits routing in high-speed networks. A basic technical problem here is to interconnect certain prescribed “channels” on the chip, and wires belonging to different sets of pins should not touch each other. In this simplest form, the problem mathematically amounts to finding disjoint trees in a graph or disjoint paths in a graph, each connecting a given set of vertices.

Fortune, Hopcroft and Wyllie [10] proved that the directed version of the problem (DDPP) is NP-hard even if  $k = 2$ , whereas the problem can be solved in polynomial time when the given digraph is acyclic and  $k$  is fixed. However, it was shown that the disjoint paths problem in undirected graphs (DPP) is solvable in polynomial time when  $k = 2$  [24,25,26].

Perhaps the biggest achievement in this area is Robertson and Seymour’s polynomial time algorithm (actually  $O(n^3)$  algorithm, where  $n$  is the number of vertices of the graph) for the disjoint paths problem when the number of terminals,  $k$ , is fixed. Actually, this algorithm is one of the spin-offs of their seminal work on Graph Minor project, spanning 23 papers, and giving several deep and profound results and techniques in Discrete Mathematics. The time complexity was improved by Reed, who gave an  $O(n^2)$  algorithm for the disjoint paths problem.

On the other hand, Schrijver [22] gave a polynomial time algorithm for the DDPP when  $G$  is a directed planar graph and  $k$  is fixed. For the DPP for planar graphs, Reed et al. [17] gave a linear time algorithm for fixed  $k$ .

We summarize the known results on the problem in Table 1 (see [23] for more results).

## 1.2 Induced Disjoint Paths Problem

As a generalization of the disjoint paths problem, we introduce a new problem called  $\nu_1, \nu_2, \dots, \nu_k$ . Let  $G$  be a graph and  $P_1, \dots, P_k$  be

**Table 1.** Complexity of DDPP and DPP

	DDPP	DPP
$k$ : constant	NP-hard (Planar digraph : P [22]) (Acyclic digraph : P [10])	P [20] (Planar graph : Linear [16,17])
$k$ : variable	NP-hard (see [11]) (Planar digraph : NP-hard [14]) (Acyclic digraph : NP-hard [7])	NP-hard (see [11]) (Planar graph: NP-hard [14])

connected subgraphs in  $G$ . We say that  $P_1, \dots, P_k$  are *induced* if  $P_i$  and  $P_j$  have neither common vertices nor adjacent vertices for any distinct  $i, j$ . In other words,  $P_1, \dots, P_k$  are induced if the following two conditions hold:

- Any pair of subgraphs have no common vertices.
- Let  $H$  be the graph obtained by contracting all edges in  $P_1, \dots, P_k$ . For each  $i = 1, \dots, k$ , let  $p_i$  be the vertex of  $H$  that corresponds to all vertices on  $P_i$ . Then  $\{p_1, p_2, \dots, p_k\}$  is a stable set in  $H$ .

We note that even if  $P_1, \dots, P_k$  are induced each  $P_i$  is not necessarily induced by some vertex set. The induced disjoint paths problem is the following problem.

**Induced disjoint paths problem (IDPP)**

**Input:** A graph  $G = (V, E)$  and a collection of vertex pairs  $\{(s_1, t_1), \dots, (s_k, t_k)\}$ .  
**Output:** Induced disjoint paths  $P_1, \dots, P_k$  in  $G$ , where  $P_i$  is a path whose end vertices are  $s_i$  and  $t_i$  for each  $i = 1, \dots, k$ .

For a digraph  $D = (V, A)$ , we also introduce a new problem called *induced disjoint dipaths problem*. Let  $P_1, \dots, P_k$  be dipaths in  $D$ . As with undirected graphs, we say that  $P_1, \dots, P_k$  are *induced* if  $P_i$  and  $P_j$  have neither common vertices nor adjacent vertices for any distinct  $i, j$ . The directed induced disjoint paths problem is defined as follows.

**Directed induced disjoint paths problem (DIDPP)**

**Input:** A directed graph  $D = (V, A)$  and a collection of vertex pairs  $\{(s_1, t_1), \dots, (s_k, t_k)\}$ .  
**Output:** Induced disjoint dipaths  $P_1, \dots, P_k$  in  $D$ , where  $P_i$  is a dipath from  $s_i$  to  $t_i$  for each  $i = 1, \dots, k$ .

The induced disjoint paths problem is an extension of the disjoint paths problem, because any instance of the disjoint paths problem can be reduced to an instance of the induced disjoint paths problem by subdividing every edge into two edges. Similarly, the directed induced disjoint paths problem is an extension of the directed version of the disjoint paths problem.

Our motivation for the IDPP and the DIDPP is not only an extension of the disjoint paths problem, but also detecting an induced subgraph. The latter area

has been developed in the recent years, and this is actually connected to the Strong Perfect Graph Theorem [4], and the recognition of the perfect graphs [2]. Let us give some examples. It is easy to test whether or not  $G$  has an even cycle, or an odd cycle. But it had been a long open question to test an even hole. This was finally done in [3,5,6]. As far as we are aware, the recognition problem for an odd hole is still open, although there is a polynomial time algorithm to test an odd hole or an anti odd hole in [2], which is, in fact, equivalent to the recognition of the perfectness by the Strong Perfect Graph Theorem [4]. As we see here, finding an induced subgraph is a hard problem. So it would be interesting to see whether or not the DPP or the DDPP can be extended to the induced versions. This is our second motivation. In fact, this motivation creates some of work for a similar concept “induced minor”, see [8,9].

Let us address the complexity issues concerning the IDPP and the DIDPP. By the above reduction, we see that the variants of the (directed) induced disjoint paths problem which correspond to NP-hard variants of the (directed) disjoint paths problem are NP-hard, that is, we obtain the following results:

- When  $k$  is a part of the input, the IDPP is NP-hard even if the given graph is planar.
- When  $k$  is a part of the input, the DIDPP is NP-hard even if the given digraph is acyclic or planar.
- The DIDPP is NP-hard even if  $k = 2$ .

### 1.3 Our Contribution

In this paper, we reveal the time complexity of several variants of the IDPP and the DIDPP as shown in Table 2. More precisely, we prove the following:

**Theorem 1.**

- (i) ...  $k$  ...  $G$  ...
- (ii) ...  $k$  ...  $G$  ...
- (iii) ...  $k = 2$  ...  $G$  ...
- (iv) ...  $k = 2$  ...  $G$  ...

(i) generalizes the result by Schrijver [22], while (ii) generalizes the result by Reed, Robertson, Schrijver and Seymour [17].

The rest of the paper is organized as follows. In Section 2, we prove (iii) and (iv) of Theorem 1, that is, we present NP-hardness results saying that the IDPP is NP-hard even if  $k = 2$ , and the DIDPP is NP-hard even if the given digraph is acyclic and  $k = 2$ . In Section 3, we show that the DIDPP is solvable in polynomial time when the given digraph is planar and  $k$  is fixed, which shows (i) of Theorem 1. Although this result implies that the IDPP is also solvable in polynomial time when the given graph is planar and  $k$  is fixed, this variant can be solved in linear time. In Section 4, we present this linear algorithm and shows (ii) of Theorem 1.



Here we make some remarks on the proof of (ii). In order to prove (ii), we shall consider somewhat more general problem, which is called “ $k$ -IDPP”. Non-induced version of this problem was considered by Reed [16]. We would like to use the method in [16,17], but there is one issue here. We need to prove that a given linkage must be induced. This makes a difference, and in order to apply the method [16,17], we need to prove the induced version of the theorems in [16]. This is our main challenge in the proof of (ii), and the most of the proof of (ii) is devoted to prove these results. Some graph minor tools are necessary. In fact, we need to generalize some of the results in Graph Minors VII [18] to the induced version. Some of the proofs there work for our case too, but some need to be extended to the induced version.

Before going to details, let us mention some notations.

**Table 2.** Complexity of DIDPP and IDPP

	DIDPP	IDPP
$k$ : constant	NP-hard (Planar digraph : P) (Acyclic digraph : NP-hard)	NP-hard (Planar graph : Linear)
$k$ : variable	NP-hard (Planar digraph : NP-hard) (Acyclic digraph : NP-hard)	NP-hard (Planar graph: NP-hard)

### 1.4 Basic Notation

For an undirected graph (or simply a graph)  $G = (V, E)$ , let  $uv$  denote an edge connecting  $u$  and  $v$ . For  $V' \subseteq V$ , the  $(V', E')$  is a subgraph  $G' = (V', E')$ , where  $E'$  consists of all edges of  $G$  spanned by  $V'$ .

For a directed graph (or a digraph)  $D = (V, A)$ , let  $(u, v)$  denote an arc which starts in  $u$  and ends in  $v$ , and for an arc  $a = (u, v)$  we define  $a^{-1} = (v, u)$ . For vertices  $v_0, v_1, \dots, v_l$  and arcs  $a_1, \dots, a_l$ , a sequence  $P = (v_0, a_1, v_1, a_2, \dots, a_l, v_l)$  is called a  $(v_0, v_l)$ -path (or a  $(v_0, v_l)$ -arc) if  $a_i = (v_{i-1}, v_i)$  for  $i = 1, \dots, l$ . If no confusion may arise, we sometimes denote  $P = (a_1, \dots, a_l)$  or identify  $P$  with its arc set  $\{a_1, \dots, a_l\}$ .

## 2 Hardness Results

In this section, we give two NP-hardness results. These results indicate that the (directed) induced disjoint paths problem is essentially different from the (directed) disjoint paths problem.

First, we show that the IDPP is NP-hard even if  $k = 2$ , whereas the DPP is solvable in polynomial time if  $k$  is fixed.

Given a graph  $G$ , we say that a cycle  $C$  is a  $(v_1, v_2)$ -hole (or an  $(v_1, v_2)$ -hole) if  $C$  is a cycle of  $G$  induced by some set of vertices. In other words,  $C$  is called a hole if  $C$  has no chords. The problem of finding a hole that passes through two given vertices is NP-hard [1].

**Theorem 2.** *Let  $G$  be a digraph with vertices  $u, v$  and edges incident to them. Then the problem of finding a hole that passes through  $u$  and  $v$  can be reduced to the IDPP with  $k = 2$ .*

We show that the problem of finding a hole that passes through  $u$  and  $v$  can be reduced to the IDPP with  $k = 2$ .

Let  $uu^-$  and  $uu^+$  be edges incident to  $u$ , and let  $vv^-$  and  $vv^+$  be edges incident to  $v$ . It is enough to show that the problem of finding a hole traveling  $u^-, u, u^+, v^-, v$ , and  $v^+$  in this order can be reduced to the IDPP, because the number of choices of  $u^-, u^+, v^-,$  and  $v^+$  is at most  $|V|^4$ .

Construct  $G'$  from  $G$  by replacing  $u, v$  and all edges incident to them with new vertices  $s_1, s_2, t_1, t_2$  and edges  $s_1u^+, v^-t_1, s_2v^+, u^-t_2$ . Then we can solve the original problem in  $G$  by solving the IDPP in  $G'$ , and hence the IDPP is NP-hard, even if  $k = 2$ .

We next show that the DIDPP is NP-hard even if the given digraph is acyclic and  $k = 2$ , whereas the DDPP is solvable in polynomial time when the given digraph is acyclic and  $k$  is fixed.

**Theorem 3.** *Let  $D = (V, A)$  be an acyclic digraph with  $k = 2$ .*

It is enough to show that 3-SAT can be reduced to the DIDPP in acyclic digraphs with  $k = 2$ . Let  $C_1 \wedge C_2 \wedge \dots \wedge C_m$  be an instance of 3-SAT with  $n$  variables  $x_1, \dots, x_n$ .

We construct an acyclic digraph  $D = (V, A)$  as follows (see Fig. [10](#)). Let  $W = \{w_1, \bar{w}_1, \dots, w_n, \bar{w}_n\}$  be a set of vertices, where  $w_i$  and  $\bar{w}_i$  correspond to the variable  $x_i$  for each  $i = 1, \dots, n$ . For each  $j = 1, \dots, m$ , let  $v_{j,1}, v_{j,2}, v_{j,3}$  be vertices, which correspond to the literal  $C_j$ , and define  $V_j = \{v_{j,1}, v_{j,2}, v_{j,3}\}$ . Let  $P = \{p_0, p_1, \dots, p_n\}$  and  $Q = \{q_0, q_1, \dots, q_m\}$  be sets of vertices, and define the vertex set  $V$  by  $V = W \cup (\bigcup_{j=1}^m V_j) \cup P \cup Q$ .

Define arc sets  $A_p$  and  $A_q$  by

$$A_p = \{(p_{i-1}, w_i), (p_{i-1}, \bar{w}_i), (w_i, p_i), (\bar{w}_i, p_i) \mid i = 1, \dots, n\},$$

$$A_q = \{(q_{j-1}, v_{j,i}), (v_{j,i}, q_j) \mid j = 1, \dots, m, i = 1, 2, 3\},$$

and let  $A_x$  be the arc set defined as follows:  $(\bar{w}_i, v_{j,l}) \in A_x$  if the  $l$ -th element of  $C_j$  is  $x_i$  and  $(w_i, v_{j,l}) \in A_x$  if the  $l$ -th element of  $C_j$  is  $\bar{x}_i$ . The arc set  $A$  is defined by  $A = A_p \cup A_q \cup A_x$ .

We now show that solving the DIDPP in  $D = (V, A)$  with respect to  $s_1 = p_0, t_1 = p_n, s_2 = q_0$ , and  $t_2 = q_m$  is equivalent to solving the original 3-SAT.

For each  $i = 1, \dots, n$ , every dipath from  $s_1$  to  $t_1$  goes through exactly one of  $w_i$  and  $\bar{w}_i$ . Then, we can see with the following observation that assigning “true” or “false” to  $x_j$  in the 3-SAT problem corresponds to deciding that  $P_1$  goes through  $w_i$  or  $\bar{w}_i$  in the DIDPP, respectively.

Suppose that the  $l$ -th element of  $C_j$  is  $x_i$  (resp.  $\bar{x}_i$ ). Then, if we assign “true” (resp. “false”) to  $x_i$  then  $C_j$  is satisfied in 3-SAT, which corresponds to the fact that if  $P_1$  goes through  $w_i$  (resp.  $\bar{w}_i$ ) then  $P_2$  can go through  $v_{j,l}$  from  $q_{j-1}$  to  $q_j$

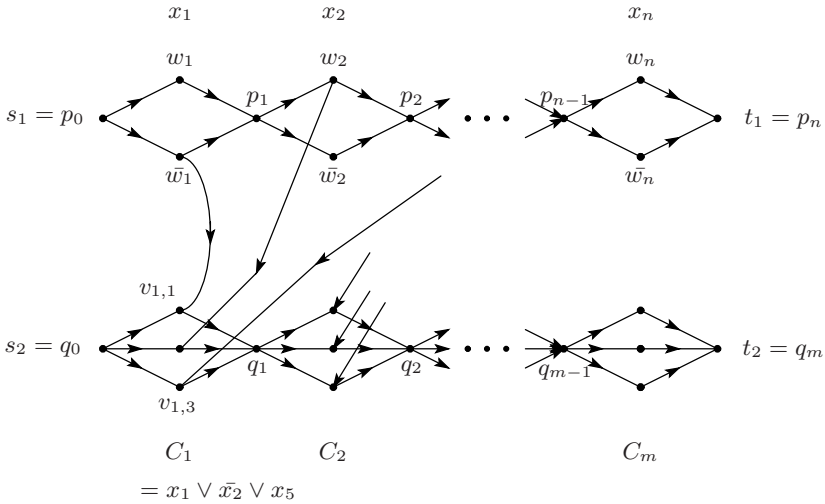


Fig. 1. Construction of  $D$

in the DIDPP. Thus,  $C_j$  is satisfied for every  $j = 1, \dots, m$  in the 3-SAT problem if and only if  $P_2$  can go from  $q_0$  to  $q_m$  in the DIDPP.

By the above arguments, 3-SAT can be reduced to the DIDPP in an acyclic digraph with  $k = 2$ .

### 3 Polynomial Time Algorithm for the DIDPP in Planar Graphs

When the given digraph  $D$  is planar and  $k$  is fixed, Schrijver gave a polynomial time algorithm for the directed disjoint paths problem [22]. As a generalization of this result, when the given digraph  $D$  is planar and  $k$  is fixed we give a polynomial time algorithm for the directed induced disjoint paths problem.

**Theorem 4.** Let  $D = (V, A)$  be a directed graph with  $k$  sources and  $k$  sinks.

Before giving the proof of Theorem 4, we show that a polynomial time algorithm for the undirected version is obtained from Theorem 4.

**Corollary 1.** Let  $G = (V, E)$  be an undirected graph with  $k$  sources and  $k$  sinks.

Consider an instance of the induced disjoint paths problem in a planar undirected graph  $G = (V, E)$ . Let  $D = (V, A)$  be the directed graph obtained from  $G$  by replacing every edge  $uv \in E$  with two arcs  $(u, v)$  and  $(v, u)$ . Then solving the induced disjoint paths problem in  $G$  corresponds to solving the directed

induced disjoint paths problem in  $D$ , and hence the original induced disjoint paths problem is solvable in polynomial time by Theorem 4.

The rest of this section is devoted to the proof of Theorem 4.

### 3.1 Preliminaries for the Proof

Let  $D = (V, A)$  be a directed planar graph, and  $\{(s_1, t_1), \dots, (s_k, t_k)\}$  be a collection of vertex pairs. The vertices  $s_1, \dots, s_k, t_1, \dots, t_k$  are called *terminals*. Without loss of generality, we assume that  $D$  is weakly connected and each terminal is incident to exactly one arc. We fix a planar embedding of  $D$ . Let  $\mathcal{F}$  be the set of all faces of  $D$ , and  $R \in \mathcal{F}$  be the unbounded face of  $D$ . For  $a \in A$ , let  $\text{left}(a)$  and  $\text{right}(a)$  be the faces in  $\mathcal{F}$  at the left-hand side and the right-hand side of  $a$ , respectively.

Let  $(G_k, \cdot)$  be the free group generated by  $g_1, g_2, \dots, g_k$ , and let 1 denote its unit element. More precisely,  $G_k$  consists of all words  $b_1 \cdots b_t$ , where  $t \geq 0$  and  $b_1, \dots, b_t \in \{g_1, g_1^{-1}, \dots, g_k, g_k^{-1}\}$  such that  $b_i b_{i+1} \neq g_j g_j^{-1}$  and  $b_i b_{i+1} \neq g_j^{-1} g_j$  for  $i = 1, \dots, t - 1$  and  $j = 1, \dots, k$ . The product  $x \cdot y$  of two words is obtained from the concatenation  $xy$  by deleting iteratively all  $g_j g_j^{-1}$  and  $g_j^{-1} g_j$ .

We say that a function  $\phi : A \rightarrow G_k$  is a *flow* if the following three conditions hold.

- For  $i = 1, \dots, k$ , the arc  $a$  leaving  $s_i$  satisfies that  $\phi(a) = g_i$ .
- For  $i = 1, \dots, k$ , the arc  $a$  entering  $t_i$  satisfies that  $\phi(a) = g_i$ .
- For each vertex  $v \in V \setminus \{s_1, \dots, s_k, t_1, \dots, t_k\}$ ,

$$\phi(a_1)^{\varepsilon_1} \cdot \phi(a_2)^{\varepsilon_2} \cdots \phi(a_l)^{\varepsilon_l} = 1,$$

where  $a_1, \dots, a_l$  are the arcs incident with  $v$ , in clockwise order, and  $\varepsilon_i = +1$  if  $a_i$  leaves  $v$  and  $\varepsilon_i = -1$  if  $a_i$  enters  $v$ .

Note that  $\phi(a)$  represents the dipaths which go through the arc  $a$ . For example, if  $\phi(a) = g_1 g_2$  then dipaths  $P_1$  and  $P_2$  go through the arc  $a$  and  $P_1$  is to the left of  $P_2$ , and if  $\phi(a) = g_3^{-1}$  then a dipath  $P_3$  goes through the arc  $a$  in the reverse direction of  $a$ . The definition of flows means that no pair of dipaths cross at any vertices. We note here the relation between directed induced disjoint paths (or directed disjoint paths) and flows. Given a solution  $\Pi = (P_1, \dots, P_k)$  of the DIDPP (or the DDPP), we define a function  $\psi_\Pi : A \rightarrow G_k$  by

$$\psi_\Pi(a) = \begin{cases} g_i & \text{if } a \text{ is an arc on } P_i, \\ 1 & \text{otherwise.} \end{cases}$$

Then  $\psi_\Pi$  is obviously a flow.

We say that two functions  $\phi, \psi : A \rightarrow G_k$  are  *$R$ -homologous* if there exists a function  $f : \mathcal{F} \rightarrow G_k$  such that

- $f(R) = 1$ ,
- $f(\text{left}(a))^{-1} \cdot \phi(a) \cdot f(\text{right}(a)) = \psi(a)$  for each arc  $a \in A$ .

It can be easily seen that if  $\phi$  is a flow and  $\psi$  is  $R$ -homologous to  $\phi$ , then  $\psi$  is also a flow.

### 3.2 Proof of Theorem 4

Schrijver’s algorithm is obtained from the following two propositions for the DDPP in an embedded planar digraph.

**Proposition 1 (Schrijver [22]).** Let  $G$  be a digraph with  $k$  terminals  $s_1, \dots, s_k$  and  $t_1, \dots, t_k$ . Let  $\phi_1, \dots, \phi_N$  be a set of  $k$  flows. Let  $\Pi$  be a solution of the DDPP. Then  $\psi_\Pi$  is  $R$ -homologous to  $\phi_1, \dots, \phi_N$ .

**Proposition 2 (Schrijver [22]).** Let  $G$  be a digraph with  $k$  terminals  $s_1, \dots, s_k$  and  $t_1, \dots, t_k$ . Let  $\phi_1, \dots, \phi_N$  be a set of  $k$  flows. Let  $\Pi$  be a solution of the DDPP. Then  $\psi_\Pi$  is  $R$ -homologous to  $\phi_1, \dots, \phi_N$ .

Proposition 1 implies the following as a corollary, because induced disjoint paths are a special case of disjoint paths.

**Proposition 3.** Let  $G$  be a digraph with  $k$  terminals  $s_1, \dots, s_k$  and  $t_1, \dots, t_k$ . Let  $\phi_1, \dots, \phi_N$  be a set of  $k$  flows. Let  $\Pi$  be a solution of the DDPP. Then  $\psi_\Pi$  is  $R$ -homologous to  $\phi_1, \dots, \phi_N$ .

For the proof of Theorem 4, we need the following proposition, which is the DIDPP version of Proposition 2.

**Proposition 4.** Let  $G$  be a digraph with  $k$  terminals  $s_1, \dots, s_k$  and  $t_1, \dots, t_k$ . Let  $\phi_1, \dots, \phi_N$  be a set of  $k$  flows. Let  $\Pi$  be a solution of the DIDPP. Then  $\psi_\Pi$  is  $R$ -homologous to  $\phi_1, \dots, \phi_N$ .

In [22], Schrijver gives a polynomial time algorithm for a problem called  $k$ -terminal flow, and proves Proposition 2 using this algorithm. Proposition 4 can be also shown with the aid of this algorithm for the CFP, but we omit the proof (see [13]).

Our algorithm for the DIDPP is obtained from Proposition 3 and Proposition 4 as follows.

**Algorithm 1.** By Proposition 3, we can find a collection of flows  $\phi_1, \dots, \phi_N$  such that for each solution  $\Pi$  of the DIDPP,  $\psi_\Pi$  is  $R$ -homologous to at least one of  $\phi_1, \dots, \phi_N$ . By Proposition 4, we can either find a solution  $\Pi$  of the DIDPP such that  $\psi_\Pi$  is  $R$ -homologous to  $\phi_i$  or conclude that such a solution does not exist, for each  $i = 1, \dots, N$ . Thus we can solve the DIDPP in polynomial time when the given digraph is planar and  $k$  is a fixed constant.

## 4 Linear Time Algorithm for the IDPP in a Planar Graph

In this section, we give a linear time algorithm that solves the IDPP in a planar graph when  $k$  is fixed. Our algorithm is based on the algorithms of [16,17] for the DPP in a planar graph.

In [16], Reed introduced a new problem called  $c$ -embedded  $k$ -realizations, which generalizes the disjoint paths problem in a planar graph, and gave a linear time algorithm for the  $c$ -embedded  $k$ -realizations. Let  $G = (V, E)$  be a graph and  $X \subseteq V$  be a vertex set whose element is called a terminal. A partition  $\mathcal{X} = \{X_1, X_2, \dots, X_p\}$  of  $X$  is  $c$ -realizable if there are disjoint trees  $T_1, \dots, T_p$  in  $G$  such that  $X_i \subseteq T_i$  for  $i = 1, \dots, p$ . Let  $\Sigma$  be a surface obtained by removing from the plane  $c$  open disks whose closures are disjoint. Such a surface is called a  $c$ -punctured plane and each disk is called a  $c$ -cuff. The boundary of  $\Sigma$  is denoted by  $bd(\Sigma)$ . The  $c$ -embedded  $k$ -realizations is described as follows.

**$c$ -embedded  $k$ -realizations**

**Input:** A graph  $G = (V, E)$  embedded on a punctured plane  $\Sigma$  with at most  $c$  cuffs, and a terminal set  $X \subseteq V \cap bd(\Sigma)$  with  $|X| = k$ .

**Output:** All realizable partitions of  $X$  in  $G$ .

Reed [16] and Reed et al. [17] gave a linear time algorithm for the  $c$ -embedded  $k$ -realizations when  $c$  and  $k$  are fixed. A linear time algorithm for the DPP in a planar graph is immediately derived from these results.

To give a linear time algorithm for the IDPP in a planar graph, we introduce the induced version of the  $c$ -embedded  $k$ -realizations. Let  $G = (V, E)$  be a graph and  $X \subseteq V$  be a terminal set. A partition  $\mathcal{X} = \{X_0, X_1, X_2, \dots, X_p\}$  of  $X$  with a special set  $X_0$  is  $c$ -induced-realizable if there are induced trees  $T_1, \dots, T_p$  in  $G$  such that  $X_i \subseteq T_i$  and  $X_0 \cap T_i = \emptyset$  for  $i = 1, \dots, p$ . We consider the following problem, which is a generalization of the induced disjoint paths problem in a planar graph.

**$c$ -embedded induced  $k$ -realizations**

**Input:** A graph  $G = (V, E)$  embedded on a punctured plane  $\Sigma$  with at most  $c$  cuffs, and a terminal set  $X \subseteq V \cap bd(\Sigma)$  with  $|X| = k$ .

**Output:** All induced-realizable partitions of  $X$  in  $G$ .

In the rest of this section, we give a linear time algorithm that solves the  $c$ -embedded induced  $k$ -realizations for any fixed  $c$  and  $k$ .

**Theorem 5.** *Let  $c$  and  $k$  be fixed. Then there is a linear time algorithm that solves the  $c$ -embedded induced  $k$ -realizations problem.*

**4.1 Deletable Vertex**

Suppose we are given an instance of the  $c$ -embedded  $k$ -realizations (resp.  $c$ -embedded induced  $k$ -realizations). We say that a vertex  $v \in V \setminus X$  is  $c$ -deletable if any realizable (resp. induced-realizable) partition  $\mathcal{X}$  of  $X$  in  $G$  is also realizable (resp. induced-realizable) in  $G - v$ . A vertex  $v \in V \setminus X$  is  $l$ -deletable if there exist  $l$  disjoint cycles  $C_1, C_2, \dots, C_l$  bounding disks  $\Delta_1, \Delta_2, \dots, \Delta_l$  such that  $v \in \Delta_1 - C_1$ ,  $\Delta_1 \subseteq \Delta_2 \subseteq \dots \subseteq \Delta_l$ , and  $\Delta_l$  does not intersect  $bd(\Sigma)$ .

The following theorem, which is a part of the main result of [21], gives a sufficient condition for a vertex to be deletable, and plays an important role in algorithms for the  $c$ -embedded  $k$ -realizations [16,17].

**Theorem 6** ([21], see [16] for a shorter proof).  $\rightarrow \dots \rightarrow k \dots \rightarrow f(k) \dots c \dots k \dots f(k) \dots$

The following theorem is the induced version of Theorem 6, which will be used in our algorithm for the  $c$ -embedded induced  $k$ -realizations.

**Theorem 7.**  $\rightarrow \dots \rightarrow k \dots \rightarrow h(k) \dots c \dots k \dots 2h(k) \dots$

Theorem 7 is one of the biggest achievements of this section, but we omit the proof due to space limitations (see [12]).

### 4.2 Algorithm

For a description of our algorithm for the  $c$ -embedded induced  $k$ -realizations, we give some preliminaries. A curve  $J \subseteq \Sigma$  is  $\rightarrow \dots \rightarrow$  if  $J \cap G \subseteq V$ , and its  $\rightarrow \dots \rightarrow$  is defined as  $|J \cap G|$ . An  $\rightarrow \dots \rightarrow$  is a proper non-self-intersecting curve in  $\Sigma$ . We say that  $J \subseteq \Sigma$  is an  $\rightarrow \dots \rightarrow$  if  $J$  is a proper non-self-intersecting (except for its end vertices) closed curve in  $\Sigma$  such that each component of  $\Sigma - J$  contains a cuff.

When  $c \geq 2$ , we can transform an instance of the  $c$ -embedded induced  $k$ -realizations into some instances with fewer cuffs by executing Algorithm Cuff\_Reduction described below. This algorithm is similar to algorithms in [16,17] for the  $c$ -embedded  $k$ -realizations, and runs in linear time.

#### Algorithm Cuff\_Reduction

**Input:** An instance of the  $c$ -embedded induced  $k$ -realizations, where  $c \geq 2$ .  
**Output:** Some instances of the  $c'$ -embedded induced  $k'$ -realizations, where  $c' < c$  and  $k'$  is at most a constant depending on  $c$  and  $k$ .

**Step 1.** If there exists an O-arc  $J$  with length at most  $8h(k) + 2$  such that each component of  $\Sigma - J$  contains at least two cuffs, then consider the inside and the outside of  $J$  separately (see Fig. 2). More precisely, let  $D_1, D_2$  be components of  $\Sigma - J$ , and consider the following two instances: one is in  $D_1 \cup J$  with terminals  $(X \cap D_1) \cup (J \cap V)$  and the other is in  $D_2 \cup J$  with terminals  $(X \cap D_2) \cup (J \cap V)$ . Then, we can reduce the instance into two instances with fewer cuffs, and stop the algorithm. We note that the solution of the original instance is obtained by unifying the solutions of two small instances in constant time.

If such O-arc does not exist, go to Step 2.

**Step 2.** If there exists an O-arc  $J$  with length at most  $8h(k) + 2$  such that one component of  $\Sigma - J$  contains exactly one cuff  $C$ , then take the shortest one among such O-arcs. If there exist some shortest O-arcs, choose such an

O-arc with the minimal bounding disk. As the same way as Step 1, we reduce the instance into two instances: one is an instance with two cuffs and the other is an instance with  $c$  cuffs in a smaller graph. For each obtained graph, execute Step 2 repeatedly, and if such O-arc does not exist in every graph, then execute Step 3 for each resulted graph.

**Step 3.** It suffices to consider the case when there is no O-arc with length at most  $8h(k) + 2$ . Denote cuffs by  $C_1, \dots, C_c$ , and find the shortest I-arc  $J_{i,j}$  connecting  $C_i$  and  $C_j$  for distinct  $1 \leq i, j \leq c$ . Let  $J$  be the shortestest I-arc among all  $J_{i,j}$ .

- 3-1. If the length of  $J$  is at most  $4h(k) + 2$ , then “open”  $\Sigma$  along  $J$  and reduce the instance into an instance with  $c - 1$  cuffs (see Fig. 3). More precisely, for each vertex  $v$  on  $J$ , split  $v$  into two vertices  $v_1, v_2$  and replace every edge  $vu$  incident to  $v$  by  $v_1u$  or  $v_2u$  so that  $J$  is contained in a new face. Furthermore, add all vertices in  $\{v_1, v_2 \mid v \in J\}$  to terminals. Then, the instance is reduced into an instance with  $c - 1$  cuffs, and stop the algorithm.
- 3-2. If the length of  $J$  is more than  $4h(k) + 2$ , delete all vertices of  $J$  except the first  $2h(k) + 1$  and the last  $2h(k) + 1$ . Then, since the length of  $J$  becomes  $4h(k) + 2$ , execute the same procedure as Step 3-1.

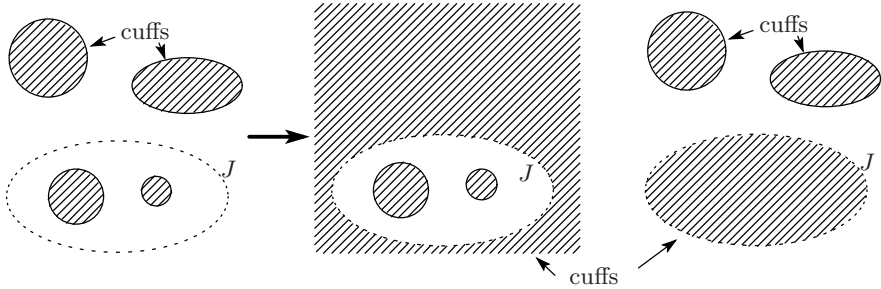


Fig. 2. Reduction to instances with fewer cuffs

To see the correctness of Algorithm Cuff\_Reduction, we prove that all vertices deleted in Step 3-2, say  $Q$ , are deletable.

**Proposition 5.** *Let  $Q$  be a set of vertices in a graph  $G$  such that  $Q$  is  $l$ -isolated and  $Q$  is not connected to the boundary of  $\Sigma$ . Then,  $Q$  is deletable.*

For a proof of this proposition, we use the following characterization of  $l$ -isolated vertices. For a vertex  $v$  in a graph  $G$ , let  $d_G(v)$  denote the minimum number of vertices of  $G$  on the interior of an I-arc  $J$ , where  $J$  is taken over all I-arcs of  $\Sigma$  with one endpoint  $v$  and the other in  $bd(\Sigma)$ . Then, the following theorem holds.

**Theorem 8 ([16,19]).** *Let  $G$  be a graph with  $2l$  terminals. If  $d_G(v) \geq l$  for every vertex  $v$  in  $G$ , then  $G$  is  $l$ -isolated.*



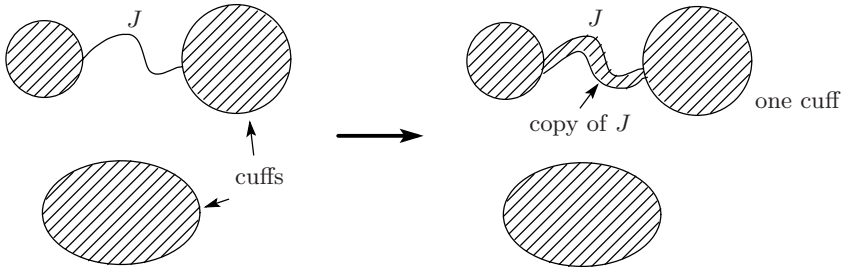


Fig. 3. “Open”  $\Sigma$  along  $J$

Now we are ready to show Proposition 5. Note that when we execute Step 3-2, we assume that the graph has no O-arc with length at most  $8h(k) + 2$  and  $J$  is the shortest I-arc with its endpoints in  $bd(\Sigma)$ .

By Theorem 7 it suffices to show that each vertex  $v$  is  $2h(k)$ -isolated in  $G - (Q \setminus \{v\})$ .

First, we show that  $G - Q$  has no O-arc with length at most  $4h(k)$ . Let  $C$  be an O-arc in  $G - Q$  with minimum length. We may assume that  $C$  intersects  $G$  only in its vertices. Then, there exist  $x, y \in Q$  such that two components  $K, K'$  of  $C - \{x, y\}$  satisfy that  $K \cap V \subseteq Q$  and  $K' \cap Q = \{x, y\}$ . Since  $J$  is the shortest I-arc, the length of  $K$  is less than or equal to that of  $K'$ . Thus, if the length of  $C$  is at most  $4h(k)$  in  $G - Q$ , then  $C$  is an O-arc in  $G$  with length at most  $8h(k) + 2$ , which contradicts the assumption.

On the other hand, as  $J$  is the shortest I-arc, we can see that  $d_{G-(Q \setminus \{v\})}(v) \geq 2h(k)$  holds for each vertex  $v \in Q$ .

Thus, by Theorem 8, each vertex  $v \in Q$  is  $2h(k)$ -isolated in  $G - (Q \setminus \{v\})$ .

**Proposition 6.**

In Steps 1 and 2, by using the augmenting path method of Ford and Fulkerson in  $G$ , we can find O-arcs with length at most  $8h(k) + 2$  in linear time. We note that the total number of finding augmenting paths is at most  $8h(k) + 3$ , which is constant. We also note that since the number of repetition of Step 2 is at most  $|V|$ , the total number of vertices increases by at most  $2(8h(k) + 2)|V|$ .

In Step 3, by adding cuffs to  $\Sigma$ , we regard  $G = (V, E)$  as a graph embedded on a plane. Let  $\mathcal{F}$  be a face set of  $G$  and  $F_1, F_2, \dots, F_c \in \mathcal{F}$  be faces containing cuffs  $C_1, C_2, \dots, C_c$ , respectively. We consider an auxiliary graph called  $\mathcal{R}$  in [15] whose vertex set is  $V \cup \mathcal{F}$  and whose edge set is

$$\{vF \mid v \in V, F \in \mathcal{F}, v \text{ is on the boundary of } F\}.$$

Then, by finding the shortest path from  $F_i$  to  $F_j$  in the radial graph for each  $1 \leq i, j \leq c$ , we can find  $J$ . Since the number of vertices in the radial graph is at most  $3|V| - 4$ , it can be done in linear time.

We note that since the number of vertices increases by at most  $2(4h(k) + 2)$  in Step 3, by executing Algorithm `Cuff_Reduction`, the total number of vertices of obtained graphs is at most  $|V| + (2(8h(k) + 2) + 2(4h(k) + 2))|V| = (24h(k) + 9)|V|$ . Thus, by repeating Algorithm `Cuff_Reduction`, we can reduce the original instance into some instances with one cuff in linear time, and the total number of vertices is at most a constant multiple of  $|V|$ . Although the original instance may be reduced into  $O(|V|)$  instances, the running time is  $O(|V|)$  in total, because the total number of vertices is  $O(|V|)$ .

When  $c = 1$ , we can determine whether a given partition  $\mathcal{X} = \{X_0, X_1, \dots, X_p\}$  is induced-realizable or not in linear time by a greedy algorithm.

As a consequence of the above arguments, we can solve the  $c$ -embedded induced  $k$ -realizations in linear time, which completes Theorem [5](#).

## Acknowledgments

The authors are thankful to Kazuo Murota, Satoru Iwata, and Kazuhisa Makino for their helpful comments on the work.

## References

1. Bienstock, D.: On the complexity of testing for even holes and induced odd paths. *Discrete Math.* 90, 85–92 (1991)
2. Chudnovsky, M., Cornuéjols, G., Liu, X., Seymour, P.D., Vuskovic, K.: Recognizing Berge graphs. *Combinatorica* 25, 143–186 (2005)
3. Chudnovsky, M., Kawarabayashi, K., Seymour, P.D.: Detecting even holes. *J. Graph Theory* 48, 85–111 (2005)
4. Chudnovsky, M., Robertson, N., Seymour, P.D., Thomas, R.: The strong perfect theorem. *Annals of Mathematics* 64, 51–219 (2006)
5. Cornuéjols, G., Conforti, M., Kapoor, A., Vuskovic, K.: Even-hole-free graphs I. Decomposition theorem. *J. Graph Theory* 39, 6–49 (2002)
6. Cornuéjols, G., Conforti, M., Kapoor, A., Vuskovic, K.: Even-hole-free graphs. II. Recognition algorithm. *J. Graph Theory* 40, 238–266 (2002)
7. Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing* 5, 691–703 (1976)
8. Fellows, M.R.: The Robertson-Seymour Theorems: a survey of applications. In: *Contemporary Mathematics*, vol. 89, pp. 1–18. American Mathematical Society (1987)
9. Fellows, M.R., Kratochvil, J., Middendorf, M., Pfeiffer, F.: The complexity of induced minors and related problems. *Algorithmica* 13, 266–282 (1995)
10. Fortune, S., Hopcroft, J., Wyllie, J.: The directed subgraph homeomorphism problem. *Theoretical Computer Science* 10, 111–121 (1980)
11. Karp, R.M.: On the computational complexity of combinatorial problems. *Networks* 5, 45–68 (1975)
12. Kawarabayashi, K., Kobayashi, Y.: A linear time algorithm for the induced disjoint paths problem in planar graphs (manuscript)
13. Kobayashi, Y.: An extension of the disjoint paths problem, METR 2007-14, Department of Mathematical Informatics, University of Tokyo (2006)

14. Lynch, J.F.: The equivalence of theorem proving and the interconnection problem. SIGDA Newsletter 5(3), 31–36 (1975)
15. Mohar, B., Thomassen, C.: Graphs on Surfaces. The Johns Hopkins University Press, Baltimore, London (2001)
16. Reed, B.: Rooted routing in the plane. Discrete Applied Math. 57, 213–227 (1995)
17. Reed, B.A., Robertson, N., Schrijver, A., Seymour, P.D.: Finding disjoint trees in planar graphs in linear time. In: Contemporary Mathematics, vol. 147, pp. 295–301. American Mathematical Society (1993)
18. Robertson, N., Seymour, P.D.: Graph minors. VII. Disjoint paths on a surface. Journal of Combinatorial Theory Ser. B 45, 212–254 (1988)
19. Robertson, N., Seymour, P.D.: Graph minors. XI. Circuits on a surface. Journal of Combinatorial Theory Ser. B 60, 72–106 (1994)
20. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. Journal of Combinatorial Theory Ser. B 63, 65–110 (1995)
21. Robertson, N., Seymour, P.D.: Graph minors. XXII. Irrelevant vertices in linkage problems (manuscript)
22. Schrijver, A.: Finding  $k$  disjoint paths in a directed planar graph. SIAM Journal on Computing 23, 780–788 (1994)
23. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Heidelberg (2003)
24. Seymour, P.D.: Disjoint paths in graphs. Discrete Mathematics 29, 293–309 (1980)
25. Shiloach, Y.: A polynomial solution to the undirected two paths problem. Journal of the ACM 27, 445–456 (1980)
26. Thomassen, C.: 2-linked graphs. European Journal of Combinatorics 1, 371–378 (1980)

# A Weighted $K_{t,t}$ -Free $t$ -Factor Algorithm for Bipartite Graphs

Kenjiro Takazawa

University of Tokyo, Tokyo 113-8656, Japan, and  
Kyoto University, Kyoto 606-8502, Japan  
takazawa@misojiro.t.u-tokyo.ac.jp

**Abstract.** For a simple bipartite graph and an integer  $t \geq 2$ , we consider the problem of finding a minimum-weight  $t$ -factor under the restriction that it contains no complete bipartite graph  $K_{t,t}$  as a subgraph. When  $t = 2$ , this problem amounts to the minimum-weight square-free 2-factor problem in a bipartite graph, which is NP-hard. We propose, however, a strongly polynomial algorithm for a certain case where the weight vector is vertex-induced on any subgraph isomorphic to  $K_{t,t}$ . The algorithm adapts the unweighted algorithms of Hartvigsen and Pap, and a primal-dual approach to the minimum-cost flow problem. The algorithm is fully combinatorial, and thus provides a dual integrality theorem, which is tantamount to Makai's theorem dealing with maximum-weight  $K_{t,t}$ -free  $t$ -matchings.

## 1 Introduction

Let  $G = (V, E)$  be a simple undirected graph, that is,  $G$  has neither parallel edges nor self-loops. Throughout this paper, we assume that the given graphs are simple. For a vector  $b \in \mathbf{Z}_+^V$ , an edge set  $M \subseteq E$  is said to be a  $b$ -matching/factor if every vertex  $v \in V$  is incident to at most  $b(v)$  edges in  $M$ , and a  $b$ -matching/factor if every vertex  $v \in V$  is incident to exactly  $b(v)$  edges in  $M$ . If  $b(v) = t$  for every  $v \in V$ , we simply refer to  $b$ -matchings/factors as  $t$ -matchings/factors.

Let us denote a cycle of length  $k$  by  $C_k$ . For a  $b$ -matching/factor  $M$  with  $b(v) \leq 2$  for each  $v \in V$ , we say that  $M$  is  $C_k$ -free if  $M$  contains no cycles of length  $k$  or less. The  $C_k$ -free 2-factor problem is to find a  $C_k$ -free 2-factor in a given graph. Note that the case where  $k \leq 2$  is exactly the classical simple 2-factor problem, which can be solved efficiently.

One important aspect of the  $C_k$ -free 2-factor problem is that it is a relaxation of the Hamilton cycle problem. From this point of view, it is easily seen that this problem is NP-hard when  $|V|/2 \leq k \leq |V| - 1$ . Moreover, Papadimitriou showed that the problem is NP-hard when  $k \geq 5$  (see [1]). On the other hand, for the case where  $k = 3$ , an augmenting path algorithm is given by Hartvigsen [2]. The  $C_4$ -free 2-factor problem is left open.

The weighted  $C_k$ -free 2-factor problem is to find a  $C_k$ -free 2-factor that minimizes the total weight of its edges for a given weighted graph. The problem is NP-hard when  $k \geq 5$ , which follows from the NP-hardness of the unweighted

problem, and so is the case where  $k = 4$  [23]. The weighted  $C_3$ -free 2-factor problem is unsettled. Polyhedral structures of  $C_k$ -free 2-factors are studied in Cunningham and Wang [3]. Related works also appear in [2,14,20].

We now focus on bipartite graphs. Note that it suffices to consider the cases where  $k$  is even. While the  $C_6$ -free 2-factor problem in bipartite graphs is NP-hard [10],  $C_4$ -free 2-factors in bipartite graphs are tractable and studied actively. We remark that a  $C_4$ -free 2-matching/factor in a bipartite graph, which is a 2-matching/factor that does not contain  $C_4$ , is often referred to as a square-free 2-matching/factor. Since the complement of an  $(n - 3)$ -connected bipartite graph is a square-free 2-matching, the theory of square-free 2-matching can be applied to the vertex-connectivity augmentation problem.

The first result on square-free 2-matchings was due to Hartvigsen [12], who proposed a characterization of graphs that admit square-free 2-factors and a combinatorial algorithm for finding one. This was followed by a min-max formula by Z. Király [15]. Then Hartvigsen [13], the journal version of [12], presented a full description of his algorithm and a constructive proof for the min-max formula.

As for the weighted  $C_k$ -free 2-factor problem in bipartite graphs, the NP-hardness when  $k \geq 6$  follows from that of the unweighted problem. Moreover, Z. Király proved that the weighted square-free 2-factor problem is also NP-hard (see [6]).

An attractive generalization of the square-free 2-factor problem is the  $K_{t,t}$ -free  $t$ -factor problem, proposed by Frank [6]. In a bipartite graph, a  $t$ -matching/factor is said to be  $K_{t,t}$ -free if it contains no  $K_{t,t}$  as a subgraph. Note that the case where  $t = 2$  is exactly the square-free 2-factor problem. Using a general framework of Frank and Jordán [7] on covering crossing bi-supermodular functions on pairs of sets, Frank [6] provided a min-max formula for  $K_{t,t}$ -free  $t$ -matchings, which extends Z. Király's formula [15] for the special case of  $t = 2$ . Through this approach, one can compute the size of the maximum  $K_{t,t}$ -free  $t$ -matching in polynomial time by the ellipsoid method or a combinatorial method by Fleiner [5]. Moreover, one can find a maximum  $K_{t,t}$ -free  $t$ -matching combinatorially by applying Végő and Benczúr's algorithm for covering pairs of sets [22]. A direct approach to this problem was done by Pap [17,18,19]. He gave a combinatorial proof for Frank's min-max formula, which implies a polynomial-time algorithm. We remark that applying Pap's algorithm to the case when  $t = 2$  results in an algorithm different from Hartvigsen's algorithm.

The weighted  $K_{t,t}$ -free  $t$ -factor problem in bipartite graphs has also been considered. As mentioned, this problem is NP-hard even when  $t = 2$ . However, Makai [16] showed a linear programming description of maximum-weight  $K_{t,t}$ -free  $t$ -matchings and proved its dual integrality for a certain class of weight vectors called  $w$ -vertex-induced. For a weight vector  $w \in \mathbf{R}^E$  and a subgraph  $H$  of  $G$ ,  $w$  is said to be vertex-induced on  $H$  if there exists a function  $\pi_H : V(H) \rightarrow \mathbf{R}$  such that  $w(uv) = \pi_H(u) + \pi_H(v)$  for every  $uv \in E(H)$ . Here,  $V(H)$  and  $E(H)$  denote the vertex set and edge set of  $H$ , respectively, and  $uv$  denotes

an edge connecting  $u, v \in V(H)$ . The class considered by Makai [16] is that  $w$  is vertex-induced on any subgraph isomorphic to  $K_{t,t}$ . Applying the ellipsoid method to Makai’s description, one obtains a polynomial algorithm for this class of weighted bipartite graphs, which could be made strongly polynomial by Frank and Tardos’ method [8].

This paper presents a combinatorial primal-dual algorithm to find a minimum-weight  $K_{t,t}$ -free  $t$ -factor in a weighted bipartite graph whose weight vector is vertex-induced on any subgraph isomorphic to  $K_{t,t}$ . The primal part of the algorithm is a variant of Hartvigsen’s and Pap’s algorithms, while the dual part is based on the framework of a primal-dual approach to the minimum-cost flow problem [4,21]. The algorithm is fully combinatorial, so the output of the algorithm is integer if the weight vector is integer. Thus, the algorithm implies dual integrality of an LP-formulation for the problem, which corresponds to Makai’s theorem [16]. The complexity of the algorithm is  $O(tn^2D)$ , where  $n$  is the number of vertices and  $D$  is the time to execute a shortest path algorithm with nonnegative length. Incorporating Fredman and Tarjan’s implementation of Dijkstra’s algorithm [9], we get a strongly polynomial complexity  $O(tn^2m + tn^3 \log n)$ , where  $m$  is the number of edges.

This paper is organized as follows. Section 2 describes a maximum-cardinality square-free 2-matching algorithm, and Sect. 3 extends it to a minimum-weight square-free 2-factor algorithm. Section 4 provides a further extension of the algorithm to the weighted  $K_{t,t}$ -free  $t$ -factor problem. Finally, Sect. 5 discusses the relation between minimum-weight  $K_{t,t}$ -free  $t$ -factors and maximum-weight  $K_{t,t}$ -free  $t$ -matchings.

Before closing this section, let us prepare some notations and definitions used in the following sections. Let  $G = (V, E)$  be an undirected graph with vertex set  $V$  and edge set  $E$ . An edge connecting  $u, v \in V$  is denoted by  $uv$ . For a vertex  $v \in V$ ,  $\delta v \subseteq E$  denotes the set of edges incident to  $v$ . For  $Z \subseteq V$ , the subgraph induced by  $Z$  is denoted by  $G[Z] = (Z, E[Z])$ , that is,  $E[Z] = \{uv \mid u, v \in Z, uv \in E\}$ . For a subgraph  $H$  of  $G$ ,  $V(H)$  and  $E(H)$  denote the vertex set and edge set of  $H$ , respectively, i.e.,  $H = (V(H), E(H))$ . A cycle is a subgraph  $(\{v_1, \dots, v_k\}, \{e_1, \dots, e_k\})$  where  $v_i \neq v_j$  if  $i \neq j$ ,  $e_i = v_i v_{i+1}$  for  $i = 1, \dots, k - 1$  and  $e_k = v_k v_1$ . A cycle consisting of  $k$  edges is denoted by  $C_k$ .

When we denote a graph by  $G = (U, V; E)$ , we mean that  $G$  is bipartite, that is, the vertex set and edge set of  $G$  are  $U \cup V$  and  $E$ , respectively, and  $E[U]$  and  $E[V]$  are empty. For subgraph  $H$  of  $G$ ,  $U(H)$  (resp.  $V(H)$ ) denotes the set of vertices in  $U$  (resp.  $V$ ) that belong to  $H$ . A complete bipartite graph  $K_{s,t}$  is a simple bipartite graph  $(U, V; E)$  with  $|U| = s$ ,  $|V| = t$  and  $E = \{uv \mid u \in U, v \in V\}$ . Recall that  $K_{2,2}$  is isomorphic to  $C_4$ , and is often called a  $\dots$ . For a subgraph  $H$  of  $G$ , a component in  $H$  isomorphic to  $K_{2,2}$  is called a  $\dots$ .  $\dots$  and the number of square-components in  $H$  is denoted by  $c(H)$ . For a bipartite graph  $G$ , let  $\mathcal{S}_t$  denote the family of all its subgraphs isomorphic to  $K_{t,t}$ . We often abbreviate  $\mathcal{S}_2$  as  $\mathcal{S}$ .

For a directed graph  $G = (V, A)$  with vertex set  $V$  and edge set  $A$ , we denote an edge  $e$  from  $u$  to  $v$  by  $uv$ , as far as it causes no confusion whether  $e$  is

directed or undirected. For  $e = uv \in A$ , the initial and terminal vertex of  $e$  are denoted by  $\partial^+e$  and  $\partial^-e$ , respectively, that is,  $\partial^+e = u$  and  $\partial^-e = v$ . A path is a subgraph  $(\{v_1, \dots, v_k\}, \{e_1, \dots, e_{k-1}\})$  where  $v_i \neq v_j$  if  $i \neq j$  and  $e_i = v_i v_{i+1}$  for  $i = 1, \dots, k - 1$ .

For two sets  $F_1, F_2 \subseteq E$ , the symmetric difference  $(F_1 \setminus F_2) \cup (F_2 \setminus F_1)$  is denoted by  $F_1 \Delta F_2$ . For a vector  $x \in \mathbf{R}^E$  and  $F \subseteq E$ , define  $x(F) = \sum_{e \in F} x(e)$ .

## 2 A Maximum Square-Free 2-Matching Algorithm

This section describes an algorithm to find a maximum square-free 2-matching in bipartite graphs. The algorithm is based on algorithms of Hartvigsen [12,13] and Pap [17,18,19], but different from both. Our algorithm uses the shortest augmenting path, whereas Pap’s algorithm does not involve the length of augmenting paths. Using the shortest path yields some simplicity, especially in the shrinking procedure, which makes the algorithm suitable for a weighted extension.

Let  $G = (U, V; E)$  be a bipartite graph and  $M \subseteq E$  be a square-free 2-matching in  $G$ . First, construct an auxiliary directed graph  $G_M = (U, V; A)$  in the following manner. Define the directed edge set  $A$  by

$$A = \{uv \mid u \in U, v \in V, uv \in E \setminus M\} \cup \{vu \mid v \in V, u \in U, uv \in M\}.$$

Where it causes no confusion, we identify the undirected edge  $uv$  in  $G$  and the directed edge  $uv$  (or  $vu$ ) in  $G_M$ . We also define two subsets  $U^\circ \subseteq U$  and  $V^\circ \subseteq V$  by  $U^\circ = \{u \mid u \in U, |\delta u \cap M| < 2\}$ ,  $V^\circ = \{v \mid v \in V, |\delta v \cap M| < 2\}$ .

Then, find a shortest path  $P$  from  $U^\circ$  to  $V^\circ$  and consider the edge set  $M' = M \Delta E(P)$ . Observe that  $M'$  is a 2-matching with  $|M'| = |M| + 1$ . Hence, if  $M'$  is square-free, then  $M'$  is a larger square-free 2-matching. We refer to the procedure to obtain  $M'$  as an *augmenting step*.

What if, however,  $M'$  contains squares? Suppose  $M \Delta E(P)$  contains a square  $S$ . Since  $P$  is the shortest  $U^\circ$ - $V^\circ$  path, we have that  $|M \cap E(S)| = 3$ , a detailed discussion of which will appear in Proposition 1. Denote  $U(S) = \{u_1, u_2\}$ ,  $V(S) = \{v_1, v_2\}$  and  $\{u_1 v_1\} = E(P) \cap E(S)$  (see Fig. 1). Then, what we do is to “shrink”  $S$ . Identify  $u_1$  and  $u_2$  to obtain a new vertex  $u_S$ , and  $v_1$  and  $v_2$  to obtain a new vertex  $v_S$ . Then, delete all edges in  $E(S)$  and connect  $u_S$  and  $v_S$  by an  $M$ -edge. If an edge in  $E \setminus E(S)$  had been incident to  $u_1$  or  $u_2$  (resp.  $v_1$  or  $v_2$ ), the edge is incident to  $u_S$  (resp.  $v_S$ ) in the resulting graph. We allow parallel edges to appear in this procedure. If an edge had belonged to  $M$ , it also belongs to  $M$  in the new graph, and otherwise it does not. We denote the resulting graph by  $\tilde{G} = (\tilde{U}, \tilde{V}; \tilde{E})$  and refer to the new  $M$ -edge  $u_S v_S$  as a *shrunk edge*. Note that it follows from  $|M \cap E(S)| = 3$  that the new  $M$  is a simple 2-matching in  $\tilde{G}$  and may contain a square that includes shrunk squares.

If more than one square appears in  $M \Delta E(P)$ , we shrink the square which is “closest” to  $U^\circ$ . That is, we shrink the square whose non- $M$ -edge appears the

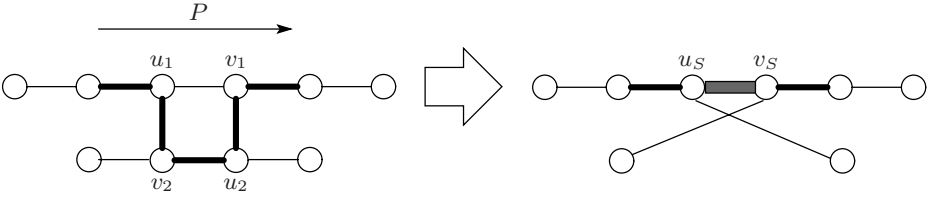


Fig. 1. Shrinking of a square (bold line:  $M$ -edge)

first in  $P$ . We refer to the procedure to obtain a new graph and 2-matching as  $\text{Shrink}(M, P)$ .

Then, we recursively execute the above procedures. Here, we have to take care that the  $U^\circ$ - $V^\circ$  path does not contain shrunk squares. In order to achieve this, we search a  $U^\circ$ - $V^\circ$  path in a subgraph  $\tilde{G}'_M$  of  $\tilde{G}_M$  obtained by deleting all the shrunk squares. Then, set  $b \in \{1, 2\}^{\tilde{U} \cup \tilde{V}}$  by

$$b(v) = \begin{cases} 1 & (v \text{ is an end vertex of a shrunk square deleted from } \tilde{G}_M), \\ 2 & (\text{otherwise}), \end{cases} \quad (1)$$

and modify the definition of  $U^\circ$  and  $V^\circ$  by

$$U^\circ = \{u \mid u \in \tilde{U}, |\delta u \cap M| < b(u)\}, \quad V^\circ = \{v \mid v \in \tilde{V}, |\delta v \cap M| < b(v)\}. \quad (2)$$

This means that what we deal with is a square-free  $b$ -matching  $M$  in  $\tilde{G}'_M$ . Thus, one would see that the shrunk squares get neither incident to each other nor nested.

After an augmentation, we expand every shrunk square to obtain the original bipartite graph  $G$ . Let  $u_S v_S$  be a shrunk square that is obtained by shrinking  $S$  with  $U(S) = \{u_1, u_2\}$  and  $V(S) = \{v_1, v_2\}$ . Now, replace the vertices  $u_S, v_S$  and edge  $u_S v_S$  by  $K_{2,2}$  induced by  $U(S) \cup V(S)$ . An edge incident to  $u_S$  or  $v_S$  is connected to a vertex in  $U(S) \cup V(S)$  to which the edge had been incident before shrinking  $S$ . Next, determine  $M$ -edges. An  $M$ -edge before expanding  $S$  also belongs to  $M$ . Then, pick up three edges in  $E(S)$  to be in  $M$  so that  $M$  forms a 2-matching. Figure 2 illustrates an example of expanding a shrunk square. By expanding every shrunk square, we obtain the original bipartite graph  $G$  and a new square-free 2-matching  $M$  of one larger size.

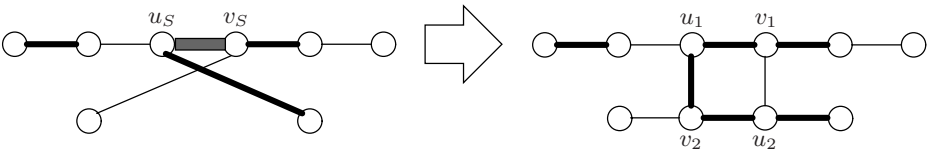


Fig. 2. Expanding of a square (bold line:  $M$ -edge)



The procedures are summarized below.

ALGORITHM MAXIMUM SQUARE-FREE 2-MATCHING

**Step 0:** Set  $M = \emptyset$  and  $\tilde{G} = G$ .

**Step 1:** If  $|M| = 2 \min\{|U|, |V|\}$ , then halt. ( $M$  is a square-free 2-factor.)

**Step 2:** Construct an auxiliary directed graph  $\tilde{G}'_M$ . In  $\tilde{G}'_M$ , define  $b$  by (1) and search for a shortest path from  $U^\circ$  to each vertex. Let  $R \subseteq \tilde{U} \cup \tilde{V}$  be the set of the reachable vertices from  $U^\circ$ . If  $V^\circ \cap R = \emptyset$ , then expand each shrunk square and halt. ( $M$  is a maximum square-free 2-matching.)

**Step 3:** Let  $P$  be the shortest path from  $U^\circ$  to  $V^\circ$ . If  $M \Delta \tilde{E}(P)$  contains a square in  $\tilde{G}'_M$ , then execute  $\text{Shrink}(M, P)$  and go to Step 2.

**Step 4:** Replace  $M$  by  $M \Delta \tilde{E}(P)$  and expand every shrunk square. Then, go to Step 1.

Here, we show that if  $M \Delta \tilde{E}(P)$  contains a square  $S$  then  $|M \cap \tilde{E}(S)| = 3$ .

**Proposition 1.**  $S \subseteq \tilde{G}'_M$ ,  $M \Delta \tilde{E}(P)$  contains a square  $S$  then  $|M \cap \tilde{E}(S)| = 3$

Since  $\tilde{E}(S) \subseteq M \Delta \tilde{E}(P)$ , the edges in  $\tilde{E}(S)$  can be partitioned into two parts,  $\tilde{E}_M = M \cap \tilde{E}(S)$  and  $\tilde{E}_P = \tilde{E}(P) \cap \tilde{E}(S)$ . We prove that  $|\tilde{E}_P| = 1$ .

As  $M$  is square-free in  $\tilde{G}'_M$ , we have that  $|\tilde{E}_P| \geq 1$ . As  $P$  visits each vertex at most once and the edges of  $M$  and  $E \setminus M$  lie alternately in  $P$ , we have that  $|\tilde{E}_P| \leq 2$ . Hence, it suffices to show that  $|\tilde{E}_P| \neq 2$ .

Denote  $U(S) = \{u_1, u_2\}$  and  $V(S) = \{v_1, v_2\}$ . To the contrary we assume, without loss of generality, that  $\tilde{E}_P = \{u_1v_1, u_2v_2\}$  and  $u_1v_1$  appears earlier than  $u_2v_2$  in  $P$ . Let us denote the subpath of  $P$  which is from the initial vertex of  $P$  to  $v_1$  by  $P_1$ , and which is from  $u_2$  to the terminal vertex of  $P$  by  $P_2$ . Here, by connecting  $P_1, u_2v_1$  and  $P_2$ , we obtain another  $U^\circ$ - $V^\circ$  path, which is shorter than  $P$ . This contradicts that  $P$  is the shortest  $U^\circ$ - $V^\circ$  path in  $\tilde{G}'_M$ .  $\square$

Now, what is left is to prove that  $M$  is maximum when the algorithm halts in Step 2. The following is a min-max formula for square-free  $b$ -matchings.

**Theorem 2** ([15]).  $G = (U, V; E)$ ,  $b \in \{0, 1, 2\}^{U \cup V}$

$$\min\{b(U \cup V \setminus Z) + |E[Z]| - c(G[Z]) \mid Z \subseteq U \cup V\}.$$

The following observation enables us to adapt Pap's proof [19] for Theorem 2 to verify the maximality of  $M$ , a detailed discussion of which is omitted.

**Proposition 3.**  $S' = \{S \mid S \in \mathcal{S}, u_S v_S \in \tilde{G}'_M, U^\circ \text{ is not in } S', S' \in \mathcal{S}'\}$

The proof is by induction on the number of shrinkings. The statement is obvious immediately after shrinking  $S$ .

Now, suppose  $P$  is a path from  $U^\circ$  to  $u_S$  that satisfies the condition in the statement and consider subsequent shrinkings. A shrinking that does not delete any edge in  $\tilde{E}(P)$  does not matter. If a shrinking deletes an edge  $e \in \tilde{E}(P)$ , then it holds that  $e \in M$ . For, if  $e \notin M$ , a square  $S_e$  appears when  $e$  turns to be an  $M$ -edge by Proposition [1](#), which contradicts that  $e$  had been in a shortest path used in shrinking  $S' \in \mathcal{S}'$  and closer to  $U^\circ$  than  $u_{S'}$ . Let  $e \in M \cap \tilde{E}(P)$  be deleted in a subsequent shrinking (an example is shown in Fig. [3](#)). Here,  $\partial^-e \in \tilde{U}$  is shrunk into  $u_{S''}$ , which is reachable from  $U^\circ$ . On the other hand,  $u_S$  is reachable from  $u_{S''}$  by tracing the subpath of  $P$  that had connected  $\partial^-e$  and  $u_S$ . Therefore, the statement is maintained in shrinking  $S''$ .  $\square$

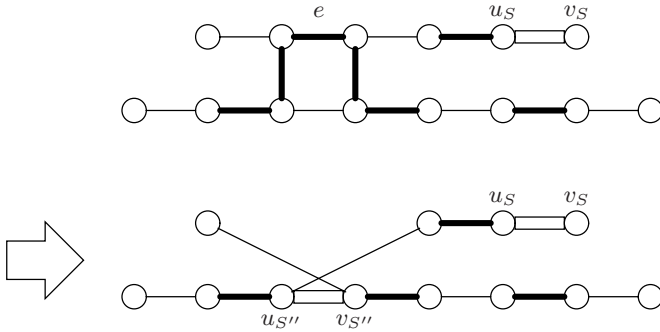


Fig. 3. Reachability of  $u_S$  (bold line:  $M$ -edge)

### 3 A Weighted Square-Free 2-Factor Algorithm

This section deals with the weighted square-free 2-factor problem. Let  $(G, w)$  be a weighted bipartite graph with  $G = (U, V; E)$  and  $w \in \mathbf{R}_+^E$ . Throughout this section, we assume that  $|U| = |V|$ . We also assume that  $w$  is vertex-induced on any square. That is, we assume that, for any square  $S$  with  $U(S) = \{u_1, u_2\}$  and  $V(S) = \{v_1, v_2\}$ , there exists a potential function  $\pi_S : U(S) \cup V(S) \rightarrow \mathbf{R}$  such that  $w(u_i v_j) = \pi_S(u_i) + \pi_S(v_j)$  for any  $i, j \in \{1, 2\}$ . In other words, it holds that  $w(u_1 v_1) + w(u_2 v_2) = w(u_1 v_2) + w(u_2 v_1)$ . We propose an algorithm to find a square-free 2-factor  $M$  that minimizes  $w(M)$  if exists, or otherwise determine that no square-free 2-factor exists in  $G$ . The algorithm, based on the primal-dual framework of the minimum-cost flow algorithm [\[4,21\]](#), extends ALGORITHM MAXIMUM SQUARE-FREE 2-MATCHING.

Let  $x \in \mathbf{R}^E$ . The following is a linear programming relaxation of an integer program for the minimum-weight square-free 2-factor problem:

$$\begin{aligned}
 \text{(P)} \quad & \text{minimize} && wx \\
 & \text{subject to} && x(\delta v) = 2 \quad (\forall v \in U \cup V), \\
 & && x(E(S)) \leq 3 \quad (\forall S \in \mathcal{S}), \\
 & && 0 \leq x(e) \leq 1 \quad (\forall e \in E).
 \end{aligned}$$

One would see that the incidence vector of a square-free 2-factor is a feasible solution for (P). In what follows, we often identify an edge set  $M$  and its incidence vector  $x$ .

Consider the dual problem of (P). Let  $p \in \mathbf{R}^{U \cup V}$ ,  $q \in \mathbf{R}^E$  and  $r \in \mathbf{R}^S$ . The dual problem is given by

$$\begin{aligned}
 \text{(D)} \quad & \text{maximize} && 2(p(U) - p(V)) - q(E) - 3r(S) \\
 & \text{subject to} && p(u) - p(v) - q(e) - \sum_{S: e \in E(S)} r(S) \leq w(e) \quad (\forall e = uv \in E), \\
 & && q, r \geq 0.
 \end{aligned}$$

The complementary slackness conditions of (P) and (D) are

$$x(e) > 0 \Rightarrow p(u) - p(v) - q(e) - \sum_{S: e \in E(S)} r(S) = w(e), \tag{3}$$

$$q(e) > 0 \Rightarrow x(e) = 1, \tag{4}$$

$$r(S) > 0 \Rightarrow x(E(S)) = 3. \tag{5}$$

In what follows, we present an algorithm to find feasible solutions for (P) and (D) which satisfy (3)–(5) by extending ALGORITHM MAXIMUM SQUARE-FREE 2-MATCHING. Roughly speaking, we maintain a square-free 2-matching  $M$ , construct an auxiliary directed graph  $\tilde{G}_M$ , search for a  $U^\circ$ - $V^\circ$  path  $P$  in its subgraph  $\tilde{G}'_M$ , and then augment  $M$  by substituting  $M \Delta \tilde{E}(P)$  for  $M$  or shrink a square. In these procedures, we also take dual solutions into account. In particular, a significant difference from ALGORITHM MAXIMUM SQUARE-FREE 2-MATCHING is that we do not expand a shrunk square  $u_S v_S$  after an augmentation if  $r(S) > 0$ , and such a shrunk square is used in the subsequent searching for a  $U^\circ$ - $V^\circ$  path.

Now, let us consider the details. Let  $\tilde{G}_M = (\tilde{U}, \tilde{V}; A)$  be an auxiliary directed graph, which may have resulted from repeated shrinking and expanding of squares. Recall that the  $M$ -edges (including all shrunk squares) are oriented in the direction from  $\tilde{V}$  to  $\tilde{U}$ , and other edges in the opposite direction. For  $\tilde{G}_M$ , a length function  $l : A \rightarrow \mathbf{R}$  is defined by

$$l(e) = \begin{cases} w(e) - p(u) + p(v) & (e \notin M \text{ and corresponds to } uv \in E), \\ -w(e) + p(u) - p(v) & (e \in M \text{ and corresponds to } uv \in E), \\ r(S) & (e \text{ is a shrunk square } u_S v_S). \end{cases}$$

Remark that  $p$  is defined on  $U \cup V$ , the vertex set of the original bipartite graph  $G$ , while  $l$  is defined on  $A$ , the edge set of  $\tilde{G}_M$ .

In the auxiliary graph  $\tilde{G}_M$ , we establish the following optimality criterion.

**Theorem 4.** Let  $M$  be a square-free 2-matching of  $G$ ,  $\tilde{G}_M$  be the auxiliary directed graph,  $p \in \mathbf{R}^{U \cup V}$ ,  $q \in \mathbf{R}^E$  and  $r \in \mathbf{R}^S$  be dual variables, and  $(G, w)$  be the weighted bipartite graph. Then,  $(M, p, q, r)$  is an optimal solution for (P) and (D) if and only if (6)–(8) hold. (6), (8)

$$r \geq 0, \quad r(S) > 0, \quad \forall S \in \mathcal{S}; \quad (6)$$

$$\forall e \in A \quad l(e) \geq 0; \quad (7)$$

$$\forall u, v \in V \quad \forall e = uv \in E(S) \quad p(u) - p(v) - r(S) = w(e). \quad (8)$$

We prove the theorem by showing how to construct feasible solutions for (P) and (D) that satisfy (3)–(5).

Let  $e = uv \in E$  be an edge not shrunk in  $\tilde{G}_M$ . If  $e \notin M$ , then by (7) we have that  $l(e) = w(e) - p(u) + p(v) \geq 0$ . Now, set  $q(e) = 0$  to have (3) and (4) hold in  $e$ . If  $e \in M$ , then  $l(e) = -w(e) + p(u) - p(v) \geq 0$  by (7). Set  $q(e) = l(e)$ , which gets (3) and (4) to hold in  $e$ .

Let  $e = uv \in E$  belong to  $E(S)$  of a shrunk square  $u_S v_S$  in  $\tilde{G}_M$ . For such  $e$ , set  $q(e) = 0$ . Then, by (8), we have that (3) and (4) hold in  $e$  regardless whether  $x(e) = 0$  or  $x(e) = 1$  after expanding  $S$ .

Now we have determined  $q(e)$  on every  $e \in E$ . From the above construction, one would appreciate the feasibility of  $(p, q, r)$  for (D). Moreover, by expanding all shrunk squares in  $\tilde{G}_M$ , we obtain a square-free 2-factor in  $G$ , a feasible solution for (P). For this pair of solutions for (P) and (D), it follows from (6) that (5) holds. Therefore, (3)–(5) hold for this pair of solutions for (P) and (D).  $\square$

Now, let us describe the minimum-weight square-free 2-factor algorithm. The algorithm keeps a square-free 2-matching  $M$  and a dual solution  $(p, r)$  that satisfy (6)–(8), and increases  $|M|$  until it attains the maximum.

#### ALGORITHM MINIMUM-WEIGHT SQUARE-FREE 2-FACTOR

**Step 0:** Set  $M = \emptyset$ ,  $p = 0$ ,  $r = 0$  and  $\tilde{G} = G$ .

**Step 1:** If  $|M| = 2|\tilde{U}|$ , then expand every shrunk square and halt. ( $M$  is a minimum-weight square-free 2-factor.)

**Step 2:** Construct an auxiliary directed graph  $\tilde{G}_M = (\tilde{U}, \tilde{V}; A)$  and delete shrunk squares that are created after the latest augmentation to obtain a new graph  $\tilde{G}'_M$ . Then, in  $\tilde{G}'_M$ , define  $b$  by (11) and search for a shortest path with respect to (w.r.t.)  $l$  from  $U^\circ$  to each vertex. Let  $R \subseteq \tilde{U} \cup \tilde{V}$  be the set of the reachable vertices from  $U^\circ$  and define  $d: \tilde{U} \cup \tilde{V} \rightarrow \mathbf{R}$  by

$$d(v) = \begin{cases} \text{distance from } U^\circ \text{ to } v \text{ w.r.t. } l & (v \in R), \\ \max\{d(v) \mid v \in R\} & (\text{otherwise}). \end{cases}$$

If  $V^\circ \cap R = \emptyset$ , then halt. (No square-free 2-factor exists.)

**Step 3:** Let  $P$  be the shortest path from  $U^\circ$  to  $V^\circ$ . If more than one shortest path exists, select a path with the minimum number of edges. If  $P$  contains a shrunk square, apply **Dual-Update** (described below), expand every shrunk square in  $P$ , and then go to Step 2.

**Step 4:** If  $M \triangle \tilde{E}(P)$  contains a square without shrunk squares, apply **Dual-Update**, execute **Dual-Shrink**( $M, P$ ) (described below), and then go to Step 2.

**Step 5:** Apply **Dual-Update**, replace  $M$  by  $M \triangle \tilde{E}(P)$ , and expand every shrunk square  $S$  with  $r(S) = 0$ . Then, go to Step 1.

We remark that a shrunk square  $u_S v_S$  with  $r(S) > 0$  is not expanded in Step 5, and belongs to  $\tilde{G}'_M$  after the augmentation.

In the procedure **Dual-Update**, we change the dual solution as follows:

$$p(v) := p(v) - d(v) \quad (v \in U \cup V),$$

$$r(S) := \begin{cases} r(S) - d(u_S) + d(v_S) & (S \text{ is shrunk}), \\ r(S) & (\text{otherwise}), \end{cases}$$

where  $d(v)$  for a vertex  $v \in (U \cup V) \setminus (\tilde{U} \cup \tilde{V})$  that is shrunk into  $v_S \in \tilde{U} \cup \tilde{V}$  is defined by  $d(v_S)$ .

The procedure **Dual-Shrink**( $M, P$ ) is twofold: update of  $p$  in two vertices; and **Shrink**( $M, P$ ). We have that  $M \Delta \tilde{E}(P)$  contains squares which does not contain shrunk squares. For such a square  $S$ , it holds that  $|M \cap \tilde{E}(S)| = 3$ , which is proven later (Proposition 5). Let  $S$  be the nearest square from  $U^\circ$  among the squares in  $M \Delta \tilde{E}(P)$ , and denote  $\tilde{U}(S) = \{u_1, u_2\}$ ,  $\tilde{V}(S) = \{v_1, v_2\}$ . Without loss of generality, we assume  $u_1 v_1 \in \tilde{E}(P) \setminus M$ . Then,  $p(u_2)$  and  $p(v_2)$  by

$$p(u_2) := p(u_2) - l(u_2 v_1), \quad p(v_2) := p(v_2) + l(u_1 v_2),$$

and call **Shrink**( $M, P$ ).

Now, let us confirm the validity of the algorithm. Note that (6)–(8) hold at the beginning of the algorithm. We prove that these conditions are maintained throughout the algorithm.

**Proposition 5.** (i) (ii)

- (i)  $l(e) \geq 0$  for all  $e \in \tilde{G}'_M$ .
- (ii) For any square  $S$  in  $M \Delta \tilde{E}(P)$ ,  $|M \cap \tilde{E}(S)| = 3$ .

We prove that (ii) holds under the assumption of (i), and (i) is maintained when (ii) holds. Then, since (i) holds at the beginning of the algorithm, (i) and (ii) inductively hold throughout the algorithm.

Let  $S$  be a square without shrunk squares such that  $\tilde{E}(S) \subseteq M \Delta \tilde{E}(P)$ . Denote  $\tilde{U}(S) = \{u_1, u_2\}$ ,  $\tilde{V}(S) = \{v_1, v_2\}$ , and  $\tilde{E}_P = \tilde{E}(P) \cap \tilde{E}(S)$ . By the argument in the proof for Proposition 1, it suffices to show that  $|\tilde{E}_P| \neq 2$  in order to prove (ii).

Assume to the contrary that  $\tilde{E}_P = \{u_1 v_1, u_2 v_2\}$  and  $u_1 v_1$  appears earlier than  $u_2 v_2$  in  $P$ . Then, it holds that  $\sum_{e \in \tilde{E}(S)} l(e) = 0$  since  $w$  is vertex-induced on  $S$ . Hence, it follows from (i) that  $l(e) = 0$  for all  $e \in \tilde{E}(S)$ . Now, as is described in the proof for Proposition 1, we have another  $U^\circ$ - $V^\circ$  path  $P'$ , which is obtained by taking  $v_1 u_2$  as a shortcut for  $P$ . It follows from (i) and  $l(v_1 u_2) = 0$  that  $P'$ , which has fewer edges than  $P$ , is no longer than  $P$  w.r.t.  $l$ . This contradicts the choice of  $P$ .

Next, we prove that (i) is maintained under the assumption of (ii). Consider **Dual-Update**. Pick up a directed edge  $e \in A$ . By the definition of  $d$ , it holds

that  $d(\partial^-e) \leq d(\partial^+e) + l(e)$ . If  $e = uv$  is in the direction from  $\tilde{U}$  to  $\tilde{V}$ , the shift of  $l(e)$  in Dual-Update is  $-(-d(u)) - d(v) = d(\partial^+e) - d(\partial^-e) \geq -l(e)$ . If  $e = vu$  is not shrunk and in the direction of  $\tilde{V}$  to  $\tilde{U}$ , i.e.,  $e \in M$ , then the shift of  $l(e)$  is  $-d(u) - (-d(v)) = -d(\partial^-e) + d(\partial^+e) \geq -l(e)$ . Finally, if  $e = v_S u_S$  is a shrunk square, the shift of  $l(e)$  is  $-d(u_S) + d(v_S) = -d(\partial^-e) + d(\partial^+e) \geq -l(e)$ . Therefore, in any case we have that  $l(e) \geq 0$  after Dual-Update. Moreover, for a shortest  $U^\circ$ - $V^\circ$  path  $P$ , the above inequalities hold with equality for each  $e \in \tilde{E}(P)$  and hence  $l(e) = 0$  after Dual-Update. Thus, in an augmentation using  $P$ , in which  $l(e)$  changes to  $-l(e)$  for  $e \in \tilde{E}(P)$ , (i) is maintained.

Consider Dual-Shrink( $M, P$ ). Since (ii) holds, the procedure Dual-Shrink( $M, P$ ) is valid. In Dual-Shrink( $M, P$ ),  $l$  changes only on the edges in  $\delta u_2 \cup \delta v_2$ . One would easily see that  $l(u_1 v_2)$  and  $l(u_2 v_1)$  becomes zero by the update of  $p(u_2)$  and  $p(v_2)$ . As we have applied Dual-Update just before Dual-Shrink( $M, P$ ), we have  $l(u_1 v_1) = 0$ . Moreover, since  $w$  is vertex-induced on  $S$ , it holds that  $l(u_2 v_2) - l(u_1 v_1) = l(u_1 v_2) + l(u_2 v_1)$ , which implies that  $l(u_2 v_2)$  also becomes zero. Meanwhile, for an edge  $e \in (\delta u_2 \cup \delta v_2) \setminus \tilde{E}(S)$ , we have that  $e \notin M$ . Hence, the shift of  $l(e)$  is equal to  $l(u_2 v_1)$  for  $e \in \delta u_2$  and equal to  $l(u_1 v_2)$  for  $e \in \delta v_2$ . Therefore,  $l(e) \geq 0$  is kept for every edge in  $\delta u_2 \cup \delta v_2$  in Dual-Shrink( $M, P$ ).  $\square$

The above argument induces the following corollaries.

**Corollary 6.** . . . Dual-Update  $l(e) = 0$  , . . .  $e \in \tilde{E}(P)$

**Corollary 7.** . . .  $S$   $l(e) = 0$  , . . .  $e \in \tilde{E}(S)$

It follows from Corollary 7 that (8) holds for  $S$  when we shrink  $S$ , which is the purpose of the update of  $p(u_2)$  and  $p(v_2)$  in Dual-Shrink( $M, P$ ).

**Proposition 8.** . . .  $u_S v_S$  . . .  $d(u_S) = 0$  . . .

It follows from Proposition 3 that  $u_S$  is reachable from  $U^\circ$  in  $\tilde{G}'_M$  by traversing edges that had been in a shortest  $U^\circ$ - $V^\circ$  path. By Corollary 6, such an edge  $e$  has its length  $l(e) = 0$  in Dual-Update executed when  $e \in \tilde{E}(P)$ , and  $l(e)$  remains to be zero until the next augmentation.  $\square$

**Proposition 9.** . . . (6), (8) , . . .

**Condition (6).** Since we change  $r(S)$  only if  $S$  is shrunk and expand  $u_S v_S$  only if  $r(S) = 0$ , it holds that  $r(S) = 0$  for every non-shrunk square  $S$ . Moreover, we have seen in Proposition 5 that  $r(S) = l(u_S v_S) \geq 0$  for every shrunk square  $u_S v_S$  in  $\tilde{G}'_M$ . As for a shrunk square  $u_S v_S$  not in  $\tilde{G}'_M$ , in other words created after the latest shrunk,  $d(u_S) = 0$  by Proposition 8, which implies  $r(S) \geq 0$  after a Dual-Update. In Dual-Shrink( $M, P$ ),  $r(S)$  is not changed since Dual-Shrink( $M, P$ ) is executed for a square containing no shrunk square.

**Condition (7).** Already proved.

**Condition (8).** By Corollary 7, (8) holds when  $S$  is shrunk. Consider the shift of  $p(u) - p(v) - r(S)$  in subsequent Dual-Update for  $e = uv \in E(S)$ . The variables are changed as follows:

$$p(u) := p(u) - d(u_S), \quad p(v) := p(v) - d(v_S), \quad r(S) := r(S) - d(u_S) + d(v_S).$$

Then,  $p(u) - p(v) - r(S)$  does not change in Dual-Update. In Dual-Shrink( $M, P$ ), the variables concerned are not changed.  $\square$

By Theorem 4 and Proposition 9, if the algorithm halts in Step 1, then we have a minimum-weight square-free 2-factor  $M$  and a dual optimal solution. If the algorithm halts in Step 2,  $G$  has no square-free 2-factor. This is shown by a similar argument as that for ALGORITHM MAXIMUM SQUARE-FREE 2-MATCHING.

Let us discuss the complexity of the algorithm. Recall that  $|U \cup V| = n$  and  $|E| = m$ . The following is an easy observation, but plays a key role in analyzing the complexity.

**Proposition 10.**

We search a  $U^\circ - V^\circ$  path  $P$  in  $\tilde{G}'_M$ , which does not contain shrunk squares created after the latest augmentation, and a shrunk square expanded by the next augmentation is contained in  $P$ .  $\square$

The bottleneck part of the algorithm lies in Step 2, determining the distance from  $U^\circ$  to every vertex. It follows from Proposition 5 that this can be computed by a shortest path algorithm with nonnegative length. By Proposition 10, we apply a shortest path algorithm  $O(n)$  times between augmentations. Since augmentations happen at most  $n$  times throughout the algorithm, the total complexity is  $O(n^2 D)$ , where  $D$  is the time to execute a shortest path algorithm with nonnegative length. Incorporating Fredman and Tarjan's version of Dijkstra's algorithm [9], we get a strongly polynomial complexity  $O(n^2 m + n^3 \log n)$ .

**Theorem 11.** ALGORITHM MINIMUM-WEIGHT SQUARE-FREE 2-FACTOR  $O(n^2 m + n^3 \log n)$ .

We should remark here that ALGORITHM MINIMUM-WEIGHT SQUARE-FREE 2-FACTOR is fully combinatorial, that is, it consists of only addition, subtraction, and comparison. Thus, the algorithm leads to the following integrality theorem.

**Theorem 12.** Let  $(G, w)$  be a bipartite graph with weight function  $w \in \mathbf{R}_+^E$ . Let  $\mathcal{S}$  be the set of all square-free 2-factors of  $G$ . Then, the following conditions (P) and (D) are equivalent:

(P)  $\exists S \in \mathcal{S} \text{ such that } r(S) > 0$

(D)  $\{S \mid S \in \mathcal{S} \text{ and } r(S) > 0\} \neq \emptyset$

## 4 Extension to $K_{t,t}$ -Free $t$ -Factors

We can naturally extend ALGORITHM MINIMUM-WEIGHT SQUARE-FREE 2-FACTOR to the minimum-weight  $K_{t,t}$ -free  $t$ -factor problem. Let  $(G, w)$

be a weighted bipartite graph with  $G = (U, V; E)$  and  $w \in \mathbf{R}_+^E$ . Assume that  $|U| = |V|$  and  $w$  is vertex-induced on any  $K_{t,t}$  in  $G$ .

Let  $x \in \mathbf{R}^E$ ,  $p \in \mathbf{R}^{U \cup V}$ ,  $q \in \mathbf{R}^E$  and  $r \in \mathbf{R}^{\mathcal{S}_t}$ . The following is a linear programming relaxation of an integer program for the minimum-weight  $K_{t,t}$ -free  $t$ -factor problem:

$$\begin{aligned}
 (\text{P}_t) \quad & \text{minimize} && wx \\
 & \text{subject to} && x(\delta v) = t \quad (\forall v \in U \cup V), \\
 & && x(E(S)) \leq t^2 - 1 \quad (\forall S \in \mathcal{S}_t), \\
 & && 0 \leq x(e) \leq 1 \quad (\forall e \in E).
 \end{aligned}$$

The dual problem of  $(\text{P}_t)$  is

$$\begin{aligned}
 (\text{D}_t) \quad & \text{maximize} && t(p(U) - p(V)) - q(E) - (t^2 - 1)r(\mathcal{S}_t) \\
 & \text{subject to} && p(u) - p(v) - q(e) - \sum_{S: e \in E(S)} r(S) \leq w(e) \quad (\forall e = uv \in E), \\
 & && q, r \geq 0.
 \end{aligned}$$

We describe an algorithm for the minimum-weight  $K_{t,t}$ -free  $t$ -factor problem by mentioning the differences from ALGORITHM MINIMUM-WEIGHT SQUARE-FREE 2-FACTOR. First, let us remark the procedure  $\text{Dual-Shrink}(M, P)$ . If a  $K_{t,t}$ , denoted by  $S$ , appears in  $M \Delta \tilde{E}(P)$ , then we have that  $|M \cap \tilde{E}(S)| = t^2 - 1$ . Let  $S$  be a  $K_{t,t}$  in  $M \Delta \tilde{E}(P)$  closest from  $U^\circ$ . Denote  $\tilde{U}(S) = \{u_1, \dots, u_t\}$  and  $\tilde{V}(S) = \{v_1, \dots, v_t\}$ , and suppose  $u_1 v_1 \notin M$ . We update the dual valuable as follows:

$$p(u_i) := p(u_i) - l(u_i v_1), \quad p(v_i) := p(v_i) + l(u_1 v_i) \quad (i = 2, \dots, t).$$

Then, we identify the vertices in  $\tilde{U}(S)$  (resp.  $\tilde{V}(S)$ ) to obtain a new vertex  $u_S$  (resp.  $v_S$ ) so that  $S$  is shrunk into a single edge  $u_S v_S$ . The edge  $u_S v_S$  belongs to  $M$  and is referred to as a  $K_{t,t}$ .

Next, in an auxiliary directed graph  $\tilde{G}'_M$ , the vector  $b$  is defined by

$$b(v) = \begin{cases} 1 & (v \text{ is an end vertex of a shrunk } K_{t,t} \text{ deleted from } \tilde{G}'_M), \\ 2 & (v \text{ is an end vertex of a shrunk } K_{t,t} \text{ in } \tilde{G}'_M), \\ t & (\text{otherwise}). \end{cases} \tag{9}$$

Then,  $U^\circ$  and  $V^\circ$  is determined by (2) according to (9).

Let us discuss the complexity. Recall that  $n = |U \cup V|$ ,  $m = |E|$  and  $D$  is the time for a shortest paths algorithm with nonnegative length. After searching for a shortest path  $P$ , we check whether  $M \Delta \tilde{E}(P)$  contains a  $K_{t,t}$ , which takes  $O(m)$  time, smaller than the complexity of a shortest path algorithm. Hence, it takes  $O(nD)$  time between augmentations. Since augmentations happen at most  $tn/2$  times, the total complexity is  $O(tn^2D)$ , which gets to  $O(tn^2m + tn^3 \log n)$  by employing  $D = O(m + n \log n)$  [9].



**Theorem 13.** *Let  $(G, w)$  be an instance of the minimum-weight  $K_{t,t}$ -free  $t$ -factor problem in bipartite graphs. Then, there is an algorithm that runs in time  $O(tn^2m + tn^3 \log n)$  and either finds a  $K_{t,t}$ -free  $t$ -factor or reports that none exists.*

As was true for ALGORITHM MINIMUM-WEIGHT SQUARE-FREE 2-FACTOR, the algorithm for the minimum-weight  $K_{t,t}$ -free  $t$ -factor problem is fully combinatorial, and thus implies the following integrality theorem.

**Theorem 14.** *Let  $(G, w)$  be an instance of the minimum-weight  $K_{t,t}$ -free  $t$ -factor problem in bipartite graphs. Let  $w \in \mathbf{R}_+^E$ . Let  $(P_t)$  and  $(D_t)$  be the following linear programs:*

$$(P_t) \quad \max \sum_{e \in E} w(e) x_e$$

$$(D_t) \quad \min \sum_{e \in E} w(e) x_e$$

*subject to  $x_e \in \{0, 1\}$  for all  $e \in E$ , and  $r(S) > 0$  for all  $S \in \mathcal{S}_t$ .*

### 5 Concluding Remarks

This paper has dealt with the minimum-weight  $K_{t,t}$ -free  $t$ -factor problem in bipartite graphs, whereas Makai [16] considered the maximum-weight  $K_{t,t}$ -free  $t$ -matching problem. Let us close this paper by mentioning that these two problems are equivalent.

In fact, the two problems are polynomially reducible to each other. Given an instance  $(G, w)$  of the minimum-weight  $K_{t,t}$ -free  $t$ -factor problem such that  $G = (U, V; E)$  and  $w$  is vertex-induced on any  $K_{t,t}$ , consider a weight vector  $w' \in \mathbf{R}^E$  defined by  $w'(e) = L - w(e)$ , where  $L$  is a sufficiently large number. Note that  $w'$  is vertex-induced on any  $K_{t,t}$  in  $G$ . Then, a maximum-weight  $K_{t,t}$ -free  $t$ -matching in  $(G, w')$  is a maximum cardinality  $K_{t,t}$ -free  $t$ -matching and hence gives a solution of the minimum-weight  $K_{t,t}$ -free  $t$ -factor problem in  $(G, w)$ .

Conversely, let us given an instance  $(G, w)$  of the maximum-weight  $K_{t,t}$ -free  $t$ -matching problem, where  $G = (U, V; E)$  and  $w$  is vertex-induced on any  $K_{t,t}$ . We also assume that  $|U| \geq t$  and  $|V| \geq t$ , as is to be expected. Then, construct a new weighted graph  $(G', w')$  as follows. If  $|U| \neq |V|$ , add isolated dummy vertices to have  $|U| = |V|$ . For any pair of vertices  $u \in U$  and  $v \in V$ , add  $2t + 2$  vertices  $u_0, u_1, \dots, u_t, v_0, v_1, \dots, v_t$  and edges

$$\{uw_0, u_0v\} \cup (\{u_i v_j \mid i \in \{0, 1, \dots, t\}, j \in \{0, 1, \dots, t\}\} \setminus \{u_0 v_0\}).$$

Then, define a new weight vector  $w'$  by

$$w'(e) = \begin{cases} L - w(e) & (e \in E), \\ L & (e: \text{new edge}), \end{cases}$$

where  $L$  is a sufficiently large number. Now, observe that  $w'$  is vertex-induced on any  $K_{t,t}$  in  $G'$  and  $G'$  admits a  $K_{t,t}$ -free  $t$ -factor. Moreover, for a minimum-weight  $K_{t,t}$ -free  $t$ -factor  $M$  in  $(G', w')$ ,  $M \cap E$  is a maximum-weight  $K_{t,t}$ -free  $t$ -matching in  $(G, w)$ .

### Acknowledgements

The author is grateful to Satoru Iwata for discussions on this topic. This work is supported by Grant-in-Aid for JSPS Fellows.

## References

1. Cornuéjols, G., Pulleyblank, W.R.: A matching problem with side conditions. *Discrete Math.* 29, 135–159 (1980)
2. Cunningham, W.H.: Matching, matroids, and extensions. *Math. Program.* 91, 515–542 (2002)
3. Cunningham, W.H., Wang, Y.: Restricted 2-factor polytopes. *Math. Program.* 87, 87–111 (2000)
4. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM* 19, 248–264 (1972)
5. Fleiner, T.: Uncrossing a family of set-pairs. *Combinatorica* 21, 145–150 (2001)
6. Frank, A.: Restricted  $t$ -matchings in bipartite graphs. *Discrete Appl. Math.* 131, 337–346 (2003)
7. Frank, A., Jordán, T.: Minimal edge-coverings of pairs of sets. *J. Comb. Theory, Ser. B* 65, 73–110 (1995)
8. Frank, A., Tardos, É.: An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica* 7, 49–65 (1987)
9. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* 34, 596–615 (1987)
10. Geelen, J. F.: The  $C_6$ -free 2-factor problem in bipartite graphs is NP-complete (unpublished, 1999)
11. Hartvigsen, D.: Extensions of Matching Theory. Ph.D. thesis, Carnegie Mellon University (1984)
12. Hartvigsen, D.: The square-free 2-factor problem in bipartite graphs. In: Cornuéjols, G., Burkard, R.E., Woeginger, G.J. (eds.) IPCO 1999. LNCS, vol. 1610, pp. 234–241. Springer, Heidelberg (1999)
13. Hartvigsen, D.: Finding maximum square-free 2-matchings in bipartite graphs. *J. Comb. Theory, Ser. B* 96, 693–705 (2006)
14. Hell, P., Kirkpatrick, D., Kratochvíl, J., Kříž, I.: On restricted two-factors. *SIAM J. Discrete Math.* 1, 472–484 (1988)
15. Király, Z.:  $C_4$ -free 2-matchings in bipartite graphs. Technical report, TR-2001-13, Egerváry Research Group (1999)
16. Makai, M.: On maximum cost  $K_{t,t}$ -free  $t$ -matchings of bipartite graphs. *SIAM J. Discrete Math.* 21, 349–360 (2007)
17. Pap, G.: Alternating paths revisited II: restricted  $b$ -matchings in bipartite graphs. Technical report, TR-2005-13, Egerváry Research Group (2005)
18. Pap, G.: A Constructive Approach to Matching and Its Generalizations. Ph.D. thesis, Eötvös Loránd University (2006)
19. Pap, G.: Combinatorial algorithms for matchings, even factors and square-free 2-factors. *Math. Program.* 110, 57–69 (2007)
20. Russell, M.: Restricted Two-Factors. Master’s thesis, University of Waterloo (2001)
21. Tomizawa, N.: On some techniques useful for solution of transportation network problems. *Networks* 1, 173–194 (1971)
22. Végh, L.A., Benczúr, A.A.: Primal-dual approach for directed vertex connectivity augmentation and generalizations. In: Proc. 16th ACM-SIAM Symposium on Discrete Algorithms, pp. 186–194. ACM-SIAM, New York (2005)
23. Vornberger, O.: Easy and hard cycle covers. Universität Paderborn (preprint, 1980)

# A New Algorithm for the Maximum Weighted Stable Set Problem in Claw-Free Graphs

Gianpaolo Oriolo<sup>1</sup>, Ugo Pietropaoli<sup>1</sup>, and Gautier Stauffer<sup>2</sup>

<sup>1</sup> Università di Roma “Tor Vergata”, Dipartimento di Ingegneria dell’Impresa,  
via del Politecnico 1, 00133 Roma, Italy

{[oriolo](mailto:oriolo@disp.uniroma2.it),[pietropaoli](mailto:pietropaoli@disp.uniroma2.it)}@disp.uniroma2.it

<sup>2</sup> IBM Research, Zurich Research Lab, Säumerstrasse 4, 8034 Rüschlikon, Switzerland  
[gst@zurich.ibm.com](mailto:gst@zurich.ibm.com)

**Abstract.** In this paper, we introduce two powerful graph reductions for the maximum weighted stable set (MWSS) in general graphs. We show that these reductions allow to reduce the MWSS in claw-free graphs to the MWSS in a class of quasi-line graphs, that we call bipolar-free. For this latter class, we provide a new algorithmic decomposition theorem running in polynomial time. We then exploit this decomposition result and our reduction tools again to transform the problem to either a single matching problem or a longest path computation in an acyclic auxiliary graph (in this latter part we use some results of Pulleyblank and Shepherd [10]). Putting all the pieces together, the main contribution of this paper is a new polynomial time algorithm for the MWSS in claw-free graphs. A rough analysis of the complexity of this algorithm gives a time bound of  $O(n^6)$ , where  $n$  is the number of vertices in the graph, and which we hope can be improved by a finer analysis. Incidentally, we prove that the MWSS problem can be solved efficiently for any class of graphs that admits a “suitable” decomposition into pieces where the MWSS is easy.

## 1 Introduction

A graph is *claw-free* if no vertex has a stable set of size three in its neighborhood. While the stable set problem is  $\mathcal{NP}$ -hard in general, it can be solved in polynomial time for claw-free graphs [8, 11, 12, 6]. The stable set problem on claw-free graphs is a fundamental generalization of the matching problem. In particular, from an algorithmic point of view, it generalizes the matching problem in two different ways, and all the current algorithms exploit either one of those two generalizations.

A first way to see this problem as a generalization of the matching problem is in terms of *augmenting paths*. Berge [1] proved that a matching  $M$  is maximal for a graph  $G$  if and only if there is no alternating path that is augmenting for  $M$ . This property, often called the *augmenting path property*, can be extended to stable sets in claw-free graphs as Berge also observed [2]. Indeed he proved that a stable set  $S$  is maximal for a claw-free graph  $G$  if and only if there is no

alternating path that is augmenting for  $S$ . The algorithms by Sbihi [11], Minty [8,9] or the variation of Minty’s algorithm given by Schrijver [12] are based on the detection of augmenting paths.

Another way to see the stable set problem in claw-free graphs as a generalization of the matching problem is in terms of *line graphs*. Indeed, *line graphs* are those graphs that can be obtained from a (possibly non simple) undirected graph  $G$  by bijectively mapping the edges of  $G$  to the vertices of a new graph  $L(G)$  and by connecting two vertices in  $L(G)$  if the corresponding edges are incident in  $G$ . It is straightforward to observe that for each vertex  $v$  in  $L(G)$ ,  $N(v)$  can be covered by two cliques. The graphs with this latter property are called *quasi-line graphs* (and thus line graphs) are claw-free. Now, due to the one-to-one mapping between the edges of a graph  $G$  and the vertices of its line graph  $L(G)$  and by definition of adjacencies in  $L(G)$ , there is also a one-to-one correspondence between matchings in  $G$  and stable sets in  $L(G)$  and thus the stable set problem in claw-free graphs is a generalization of the stable set problem in line graphs, which is a matching problem. Lovász and Plummer [6] defined graph reductions that preserve the stability number to reduce the stable set problem in a claw-free graph to an unweighted stable set problem in a line graph.

Unfortunately Lovász-Plummer’s approach does not deal with the weighted version of the stable set problem on claw-free graphs. Therefore, the only algorithm for the weighted case is the one given in 1980 by Minty [8,9] (and the slightly different version discussed in Schrijver [12]). This algorithm is not very “natural”, as it converts the original problem to the problem of detecting augmenting paths in some auxiliary graphs, called “Edmonds graphs”, on which some matching problems are solved. Moreover, its overall complexity is  $O(n^6)$ , where  $n$  denotes the number of vertices in the original graph.<sup>1</sup> The question of finding a more direct and fast algorithm arises therefore naturally, as for instance it was recently pointed out in [7] “...one needs a better reduction from weighted claw-free to weighted line graphs, which seems to be a challenging research problem.”

We also point out that an elegant algorithm has been given by Pulleyblank and Shepherd [10] for the maximum weighted stable set MWSS problem on a subclass of claw-free graphs, called *acyclic digraphs*. The algorithm is based on finding a longest path in an acyclic digraph and has complexity  $O(n^3)$ .

In this paper, we propose a new algorithm for the maximum weighted stable set (MWSS) problem in claw-free graphs. The algorithm is based on graph reductions and on a new decomposition theorem for a class of quasi-line graphs, that we call *bipolar-free*. First, we perform graph reductions that somehow extend the approach of Lovász and Plummer to the weighted case as to end up with an MWSS problem on a bipolar-free quasi-line graph. We here use our decomposition theorem stating that a bipolar-free quasi-line graph that is not distance claw-free can be decomposed into at most  $n$  distance claw-free graphs, that,

---

<sup>1</sup> In [7], it is claimed that the complexity of Minty’s algorithm is  $O(n^7)$ ; we have however not been able to convince ourselves that this is correct.

because of their properties, we call distance simplicial. When the graph is distance claw-free, we use the algorithm by Pulleyblank and Shepherd [10] and otherwise, using our decomposition, it is possible to evaluate an MWSS for the original graph by solving a matching problem. A rough analysis of the complexity of the proposed algorithm gives a time bound of  $O(n^6)$ , which we hope can be improved by a finer analysis, that we defer to the journal version of the paper. The steps of the algorithm are summarized below with pointers to the corresponding results in the paper.

### Sketch of the Algorithm

1. Check that  $G$  contains a stable set of size 4 by enumeration ( $O(n^4)$ )
  - If not, find an MWSS by enumeration in  $O(n^3)$  and stop.
2. Check that  $G$  is quasi-line (by detecting 5-wheels)
  - If not, reduce the MWSS problem on  $G$  to the MWSS problem on a graph  $G'$  which is either quasi-line or  $\alpha(G') \leq 3$  ( $O(n^4)$ , see Lemma 1.3). If  $\alpha(G') \leq 3$ , solve the problem on  $G'$  by enumeration.
3.  $G'$  is a quasi-line graph (possibly,  $G' = G$ ). Check that  $G'$  is bipolar-free (by detecting bipolar pairs)
  - If not, add appropriate edges to turn  $G'$  into a bipolar-free quasi-line graph  $G''$  with the same vertex set and  $\alpha_w(G'') = \alpha_w(G')$  ( $O(n^6)$ , see Lemma 1.7). Every stable set of  $G''$  is also a stable set of  $G'$ .
4.  $G''$  is a bipolar-free quasi-line graph (possibly,  $G'' = G'$ ). Check whether  $G''$  has strongly regular articulation cliques ( $O(n^3)$ , see Lemma 2.0)
  - If not,  $G''$  is distance claw-free: find an MWSS using the algorithm from [10] ( $O(n^3)$ )
  - else decompose  $G''$  into distance simplicial strips, evaluate the crucial stable sets for each of them, and solve a matching problem to re-combine them to an MWSS of  $G''$  ( $O(n^4)$ , see Lemmas 3.1 and 5) and therefore of  $G'$ .

The paper is organized as follows. In Section 2, we introduce two graph reductions for the MWSS in general graphs. In Section 3 we show how to use our first reduction to transform the MWSS problem on a claw-free graph to the same problem on a quasi-line graph. In Section 4 we show how to make use of our second reduction to reduce the MWSS problem on quasi-line graphs to the MWSS problem on bipolar-free quasi-line graphs. Finally, Section 5 is devoted to prove a decomposition theorem for this last class of graphs describing their structure, and to present the resulting algorithm for the MWSS on the class of claw-free graphs.

### Definitions and Notations

We close the introduction with some definitions and notations. Every graph  $G(V, E)$  will be undirected and simple, i.e. without loops and parallel edges. The graph complement of  $G$  is denoted by  $\overline{G}$ . A  $\{e_1, \dots, e_k\}$  in  $G$  is a set of edges that are pairwise non-incident. A  $\{v_1, \dots, v_k\}$  in  $G$  is a set of vertices which are pairwise non-adjacent. The  $\alpha(G)$  of  $G$  is the size of a stable set

of maximum cardinality and it is denoted by  $\alpha(G)$ . Given a weight function  $w : V \mapsto \mathbb{R}$ , the weight of an MWSS in  $G$  and is denoted by  $\alpha_w(G)$ . Given a set  $U \subseteq V$ , we denote by  $G[U]$  the graph induced by the vertices in  $U$ . When no confusion can arise, we write  $\alpha(U)$  and  $\alpha_w(U)$  instead of  $\alpha(G[U])$  and  $\alpha_w(G[U])$ , respectively. Moreover, we denote by  $w(U)$  the sum of the weights of all vertices of  $U$ . A set of pairwise adjacent vertices in  $G$  is a set of pairwise adjacent vertices. We say that a clique  $K$  is maximal if there does not exist a clique  $K' \supset K$ .

Given a vertex  $u \in V$ , we denote by  $N_G(u)$  the neighbors of  $u$  in  $G$ , i.e.  $N_G(u) := \{v \in V : (u, v) \in E\}$ . When no confusion can arise, we denote  $N_G(u)$  by  $N(u)$ . A vertex  $u \in V$  is said to be universal to  $v$  if  $u$  is adjacent to  $v$  and to every vertex in  $N(v) \setminus \{u\}$ ; we denote by  $U(v)$  the set of vertices that are universal to  $v$ . A vertex  $u \in V$  is said to be a universal vertex if  $N(u)$  is a clique. Given a set  $U \subseteq V$ , we denote by  $E(U)$  the edges with both endpoints in  $U$ , by  $\delta(U)$  the edges with one endpoint in  $U$  and the other in  $V(G) \setminus U$  and by  $N(U)$  the set  $\{v \in V \setminus U : (u, v) \in E, \text{ for some } u \in U\}$ . We also denote for all  $j \geq 2$ ,  $N_j(U) := N(N_{j-1}(U)) \setminus N_{j-2}(U)$  with the convention  $N_0(U) := U$  and  $N_1(U) := N(U)$ . A vertex  $u \in V$  is said to be a universal vertex if  $N(u)$  and  $N_2(u)$  are cliques. Given two sets  $U, U' \subseteq V$ , we denote by  $E(U : U')$  the edges with one end in  $U$  and the other in  $U'$ . Two disjoint sets of vertices  $A, B$  are said to be independent (resp. complete) if  $E(A : B) = \{(a, b) : a \in A, b \in B\}$  (resp.  $E(A : B) = \emptyset$ ).

A path  $P$  of length  $k$  in  $G(V, E)$  is an ordered sequence of edges  $(e_1, \dots, e_k)$  where  $e_i = (v_i, v_{i+1})$ ,  $v_i \in V$  for all  $i = 1, \dots, k$  and  $(v_1, v_{k+1}) \notin E$  (repetition of edges is not allowed). We denote  $P = (e_1, \dots, e_k)$ . Since we consider simple graphs, we can also define  $P$  by the ordered sequence of vertices  $(v_1, \dots, v_{k+1})$  that are visited.

A graph  $(c; v_1, v_2, v_3)$  is a graph with vertex set  $\{c, v_1, v_2, v_3\}$  and edge set  $\{(c, v_1), (c, v_2), (c, v_3)\}$ . A graph  $(c; v_1, \dots, v_5)$  is a graph with vertex set  $\{c, v_1, \dots, v_5\}$  and edge set  $\bigcup_{i=1}^5 \{(c, v_i), (v_i, v_{i+1})\}$  (with the convention  $v_6 = v_1$ ). A graph  $(u_1, u_2, u_3; v_1, v_2, v_3)$  is a graph with vertex set  $\{u_1, u_2, u_3, v_1, v_2, v_3\}$  and edge set  $\{(v_1, v_2), (v_2, v_3), (v_3, v_1), (u_1, v_1), (u_2, v_2), (u_3, v_3)\}$ . A graph  $(v_1, v_2, v_3, v_4, v_5)$  is the graph with vertex set  $\{v_1, \dots, v_5\}$  and edge set  $\{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_4), (v_3, v_4), (v_3, v_5), (v_4, v_5)\}$ .

Finally, for a set  $S$ , a family of subsets  $\{S_1, \dots, S_k\}$  is called a chain if, for  $1 \leq i < j \leq k$ ,  $S_i \cap S_j \neq \emptyset$  implies  $S_i \subseteq S_j$  or  $S_j \subseteq S_i$ .

## 2 Reductions for the Maximum Weighted Stable Set Problem

### 2.1 A Simple Reduction for Strips-Composed Graphs

Chudnovsky and Seymour [3] introduced a composition operation in order to define their decomposition result for claw-free graphs. This composition procedure is general and applies to non-claw-free graphs as well. We borrow a couple

of definitions from their work (even if our definition of  $\sim_{\phi}$  is slightly different). A graph  $(G, a, b)$  is a graph (not necessarily connected) with two designated simplicial vertices  $a$  and  $b$ . Observe that, by definition, if  $a$  and  $b$  are adjacent, then  $N(a) \cup N(b)$  is a clique.

Given two vertex-disjoint strips  $(G_1, a_1, b_1)$  and  $(G_2, a_2, b_2)$ , we define the  $\sim_{\phi}$  of those two strips as the union of  $G_1 \setminus \{a_1, b_1\}$  and  $G_2 \setminus \{a_2, b_2\}$  together with all edges between  $N_{G_1}(a_1)$  and  $N_{G_2}(a_2)$  and all edges between  $N_{G_1}(b_1)$  and  $N_{G_2}(b_2)$ . Moreover, we add all edges between  $N_{G_1}(a_1)$  and  $N_{G_1}(b_1)$  when  $a_2$  and  $b_2$  are adjacent (and vice versa). Observe that this operation is closely related to the definition of 2-join (cf. [4]). Note also that gluing  $(G_1, a_1, b_1)$  and  $(G_2, b_2, a_2)$  would not result in the same graph.

We can generalize the operation of gluing to several strips by introducing a composition operation.

**Definition 1.** A composition of the strips  $(G_1, a_1, b_1), \dots, (G_k, a_k, b_k)$  w.r.t.  $(G^0, \phi)$  is a graph  $G^k$  such that  $(a_i, b_i) \notin E(G_i)$ ,  $i = 1, \dots, k$  and  $\phi = \{a_1, \dots, a_k, b_1, \dots, b_k\}$ .  $V(G^0) = \{a_1, \dots, a_k, b_1, \dots, b_k\}$ .  $G^k$  is a composition of the strips  $(G_i, a_i, b_i)$ ,  $i = 1, \dots, k$  w.r.t.  $(G^0, \phi)$ .

It is a simple exercise to prove that the composition does not depend on the order of the strips. In the following, we will refer to a graph that, as  $G^k$ , can be obtained by this procedure as a  $\sim_{\phi}$ . We will also use the following alternative definition for strips-composed graphs.

**Definition 2.** A composition of the strips  $(G_1, a_1, b_1), \dots, (G_k, a_k, b_k)$  w.r.t. the partition  $\mathcal{P} := \{P_1, \dots, P_m\}$  is a graph  $G^k$  such that  $(a_i, b_i) \notin E(G_i)$ ,  $i = 1, \dots, k$  and  $\mathcal{P} := \{P_1, \dots, P_m\}$  is a partition of  $\{a_1, \dots, a_k, b_1, \dots, b_k\}$ .  $G^k$  is a composition of the strips  $(G_i, a_i, b_i)$ ,  $i = 1, \dots, k$  w.r.t. the partition  $\mathcal{P}$ .

It is again a simple exercise to prove that the composition does not depend on the order of the classes in the partition. It is also easy to prove that the two definitions are equivalent, i.e. they define the same class of graphs, and that we can pass easily from one representation to the other. Indeed, the disjoint cliques of  $G^0$  (together with  $\phi$ ) define a natural partition of the vertices  $\{a_1, \dots, a_k, b_1, \dots, b_k\}$  and from a partition it is easy to define a suitable set of disjoint cliques and a mapping. We skip the details. In both cases, we say that  $(G_i, a_i, b_i)$ ,  $i = 1, \dots, k$  define a  $\sim_{\phi}$ , either of  $G^k$  w.r.t.  $(G^0, \phi)$  or for  $G^m$  w.r.t.  $\mathcal{P}$ .

The different strips involved in the composition can be complex objects. Nevertheless, when we are dealing with stable set problems, there is a simple reduction that allows to get rid of non desirable strips (e.g. non-line strips).

**Lemma 3.** Let  $(G, a, b)$  and  $(H, a, b)$  be graphs such that  $(a, b) \notin E(H)$ . Let  $w : V \mapsto \mathbb{R}$  be a weight function. Let  $G'$  and  $H'$  be graphs such that  $(T, v_1, v_5)$  and  $(H', c, d)$  are strips. Let  $A := N_H(a)$



$B := N_H(b)$   $C = N_{H'}(c)$   $D = N_{H'}(d)$   $w'$   $G'$

- $w'(v) = w(v)$ ,  $v \in V(H' \setminus \{c, d\})$
- $w'(v_2) := w_{\overline{B}} - w_{\overline{A \cup B}}$
- $w'(v_3) = w_H - w_{\overline{A \cup B}}$
- $w'(v_4) := w_{\overline{A}} - w_{\overline{A \cup B}}$

$w_H$   $MWSS$   $H \setminus \{a, b\}$   $(c, d) \notin H'$   $MWSS$   $H \setminus \{a, b\}$   $A \cup B$   $(c, d) \in H'$   $w_{\overline{B}}$   $w_{\overline{A}}, w_{\overline{A \cup B}}$   $MWSS$   $H \setminus \{a, b\}$   $B$   $A \cup B$   $\alpha_{w'}(G') = \alpha_w(G) - w_{\overline{A \cup B}}$

Let  $S$  be an MWSS in  $G$ . We have to deal with four cases. Either  $S$  does not pick any vertex in  $C \cup D$  or it picks a vertex in  $D$  but not in  $C$ , or it picks a vertex in  $C$  but not in  $D$ , or it picks a vertex in  $C$  and a vertex in  $D$  (possibly the same if  $C \cap D \neq \emptyset$ ).

Let us analyse the first situation. We claim that  $w(S \cap V(H)) = w_H$ . If  $c$  and  $d$  are not adjacent,  $S \cap V(H)$  must be an MWSS in  $V(H) \setminus \{a, b\}$ : otherwise, if  $w_{S \cap V(H)} < w_H = w(S_H)$ , for some suitable stable set  $S_H \subseteq V(H) \setminus \{a, b\}$ ,  $S_H \cup (S \cap V(H'))$  would be a better stable set for  $G$  (it is a stable set, since there are no adjacencies between  $V(H)$  and  $V(H') \setminus (C \cup D)$  in  $G$  and there is no new adjacency in  $G[V(H)]$ ). If  $c$  and  $d$  are adjacent, then  $S \cap V(H)$  takes at most one vertex in  $A \cup B$  (by definition  $A$  is complete to  $B$  in  $G$  in that case) and must thus be an MWSS in  $V(H) \setminus \{a, b\}$  picking at most one vertex in  $A \cup B$  for the same reasons.

Now in  $G'$ ,  $S' = \{v_3\} \cup (S \cap V(H'))$  is a stable set (since  $S$  does not intersect  $C \cup D$ ). But  $w'(S') = w'(S \cap V(H')) + w'(v_3) = w(S \cap V(H')) + w_H - w_{\overline{A \cup B}} = w(S \cap V(H')) + w(S \cap V(H)) - w_{\overline{A \cup B}} = w(S) - w_{\overline{A \cup B}}$ . Hence  $\alpha_{w'}(G') \geq \alpha_w(G) - w_{\overline{A \cup B}}$ . The three other cases can be analyzed similarly.

Conversely, let  $S'$  be an MWSS in  $G'$ . Again we have to deal with the four cases above. Let us analyze the case where  $S'$  does not intersect  $C \cup D$ : in this case, w.l.o.g., we may assume that  $v_3 \in S'$  (since  $w'(v_4) \leq w'(v_3) \geq w'(v_2)$ ). Thus let  $S''$  be an MWSS in  $H \setminus \{a, b\}$  (picking at most one vertex in  $A \cup B$  if  $c$  and  $d$  are adjacent).  $S = S'' \cup (S' \cap V(H'))$  is a stable set in  $G$ . But  $w(S) = w(S'') + w(S' \cap V(H')) = w_H + w'(S' \cap V(H')) = w'(v_3) + w_{\overline{A \cup B}} + w'(S' \cap V(H')) = w'(S') + w_{\overline{A \cup B}}$ . Thus  $\alpha_{w'}(G') \leq \alpha_w(G) - w_{\overline{A \cup B}}$ . Again the three other cases can be analyzed similarly.  $\square$

Let  $(H, a, b)$  be a strip and let  $S_H$  be an MWSS in  $H \setminus \{a, b\}$  (picking at most one vertex in  $A \cup B$  if  $c$  and  $d$  are adjacent),  $S_{\overline{B}}$ , (resp.  $S_{\overline{A}}, S_{\overline{A \cup B}}$ ) an MWSS in  $H \setminus \{a, b\}$  not intersecting  $B$  (resp.  $A, A \cup B$ ). We say that  $S_H, S_{\overline{B}}, S_{\overline{A}}, S_{\overline{A \cup B}}$  are the MWSS for  $(H, a, b)$ .

From the proof of the above lemma not only one can get  $\alpha_w(G)$  from  $\alpha_{w'}(G')$ , but also one can build an MWSS for  $G$  from an MWSS for  $G'$  in constant time, if the crucial MWSS for  $(H, a, b)$  are given.



We have a result similar to Lemma 3 for strips-composed graphs.

**Lemma 5.** Let  $G$  be the composition of the strips  $(G_i, a_i, b_i)$ ,  $i = 1, \dots, k$  w.r.t. some pair  $(G^0, \phi)$ . Let  $\mathcal{P}$  be a family of MWSS  $\{P_i\}_{i=1}^k$  of  $G$  such that  $P_i \cap P_j = \emptyset$  for  $i \neq j$ . Let  $n = \sum_{i=1}^k p_i(n)$  and  $m = \text{match}(n)$ . Then, there exists a MWSS  $H$  of  $G$  such that  $|H| = m$  and  $\alpha_w(H) = \alpha_w(G) - \sum_{i=1}^k w_{A_i \cup B_i}$ .

First of all, as we pointed out above, we can equivalently express  $G$  as the composition of the strips  $(G_i, a_i, b_i)$ ,  $i = 1, \dots, k$ , w.r.t. some pair  $(G^0, \phi)$ . We now use the procedure defined in Lemma 3 and associate to each strip  $(G_i, a_i, b_i)$  a gem  $(T^i, v_1^i, v_5^i)$  with suitable weights. Let  $H$  be the composition of the strips  $(T^i, v_1^i, v_5^i)$ ,  $i = 1, \dots, k$  w.r.t.  $(G^0, \phi)$ . For each strip  $i$ , let  $w_{A_i \cup B_i}$  be the weight of an MWSS in  $G_i \setminus \{a_i, b_i\}$  not intersecting  $A_i \cup B_i$ , where  $A_i = N_{G_i}(a_i)$  and  $B_i = N_{G_i}(b_i)$ .

**Claim 6.**  $\alpha_w(H) = \alpha_w(G) - \sum_{i=1, \dots, k} w_{A_i \cup B_i}$

$G$  is the composition of  $(G_1, a_1, b_1), \dots, (G_k, a_k, b_k)$  w.r.t.  $(G^0, \phi)$ . Let  $G'$  be the composition of  $(G_1, a_1, b_1), \dots, (G_{k-1}, a_{k-1}, b_{k-1}), (T^k, v_1^k, v_5^k)$  w.r.t.  $(G^0, \phi)$ . It follows from Lemma 3 that  $\alpha_w(G) = \alpha_w(G') + w_{A_k \cup B_k}$ . Recall that the composition does not depend on the order of the strips, i.e.  $G'$  is also the composition of  $(G_1, a_1, b_1), \dots, (G_{k-2}, a_{k-2}, b_{k-2}), (T^k, v_1^k, v_5^k), (G_{k-1}, a_{k-1}, b_{k-1})$  w.r.t.  $(G^0, \phi)$ . We now define  $G''$  to be the composition of  $(G_1, a_1, b_1), \dots, (G_{k-2}, a_{k-2}, b_{k-2}), (T^k, v_1^k, v_5^k), (T^{k-1}, v_1^{k-1}, v_5^{k-1})$  w.r.t.  $(G^0, \phi)$ . We have that  $\alpha_w(G') = \alpha_w(G'') + w_{A_{k-1} \cup B_{k-1}}$ . The claim follows by iterating this reasoning.

**Claim 7.**  $H$  is the line graph of a graph  $F$  with  $|V(F)| = k + p \leq 3k$  and  $|E(F)| = |V(H)| = 3k$ .  $O(k)$

First, note that  $H$  has  $3k$  vertices.  $G^0$  is a graph with  $2k$  vertices that is the disjoint union of  $p$  cliques. By definition, each clique of  $G^0$  induces a clique of  $H$ . We then consider the family  $\mathcal{K}$  of cliques of  $H$  that is made of the  $p$  previous cliques together with the cliques  $\{v_2^1, v_4^1\}, \dots, \{v_2^k, v_4^k\}$ . It is easy to see that  $\mathcal{K}$  covers all the edges of  $H$  and that every vertex of  $H$  belongs to exactly 2 cliques of  $\mathcal{K}$ . It is shown in [6] that, in this case,  $H$  is the line graph of a graph  $F$ , that can be built as follows. For each clique  $K$  in  $\mathcal{K}$ , we associate a vertex  $v_K$  in  $F$ . For all vertices  $v_K \neq v_{K'}$ , we add  $|K \cap K'|$  edges  $(v_K, v_{K'})$ . Observe that we can build  $F$  directly from  $G$ , i.e. we do not need to build  $H$ , in time  $O(k)$ , since  $|V(F)| = k + p \leq 3k$  and  $|E(F)| = |V(H)| = 3k$ .

We are thus left with solving a weighted matching problem in  $F$ . The weights of the edges of  $F$  (vertices of  $H$ ) can be given in time  $O(\sum_{i=1, \dots, k} p_i(n))$ . Moreover, it follows by induction from Remark 4 that an MWSS of  $G$  can be built in time  $O(k)$  from an MWSS of  $H$ , i.e. a maximum weighted matching of  $F$ . Observe that  $k = O(n)$ . The statement follows.  $\square$

We want to stress again that Lemma 3 and Lemma 5 apply not only to claw-free graphs, but to all graphs that can be obtained as the composition of strips.

### 2.2 Semi-homogeneous Pairs of Cliques

The notion of  $(A, B)$  extend the classical concept of homogeneous pair of cliques (a pair of cliques  $(A, B)$  is homogeneous if for all  $x \in V \setminus \{A \cup B\}$ ,  $x$  is either complete to  $A \cup B$  or anti-complete to  $A \cup B$  or complete to  $A$  (resp.  $B$ ) and anti-complete to  $B$  (resp.  $A$ )).

**Definition 8.**  $(A, B)$   $G(V, E)$  semi-homogeneous  $x \in V \setminus \{A \cup B\}$   $x$   $A$   $B$   $A \cup B$

**Lemma 9.**  $G = (V, E)$   $w$   $(A, B)$   $\{a, b\}$  MWSS  $A \cup B$   $(a, b)$   $A$   $B$   $\alpha_w$   $G$

Suppose that  $S$  is an MWSS of  $G$  picking a vertex  $a'$  in  $A$  and a vertex  $b'$  in  $B$ . For all  $s \in S \setminus \{a', b'\}$ ,  $s$  is neither complete to  $A$  nor to  $B$ . Thus since  $(A, B)$  is semi-homogeneous,  $s$  is anti-complete to  $A \cup B$ . Therefore we could replace  $a'$  by  $a$  and  $b'$  by  $b$  and getting a stable set of weight as big.  $\square$

We will use the previous lemma to get rid of some “annoying” pairs of vertices in quasi-line graphs, that we call bipolar. In fact we will show that if a quasi-line graph has a bipolar pair of vertices, then it contains a semi-homogeneous pair of cliques  $(A, B)$  such that there are at least two missing edges between  $A$  and  $B$ .

## 3 From Claw-Free Graphs to Quasi-Line Graphs

Recall that a graph  $G$  is quasi-line if, for all  $v$  in  $V(G)$ ,  $N(v)$  can be covered by two cliques, that is,  $\overline{G[N(v)]}$  is bipartite. Therefore, a claw-free graph  $G$  is quasi-line if and only if there exists a vertex  $v$  with an odd-hole in  $\overline{G[N(v)]}$ .

While claw-free graphs with small stability number can be significantly different from quasi-line graphs, Fouquet [5] proved that claw-free graphs with stability number greater than 3 do not differ that much from quasi-line graphs.

**Lemma 10.** [5]  $G$   $\alpha(G) \geq 4$   $5$

We have now a couple of lemmas whose proofs we defer to the journal version of the paper. The first one deals with the gluing operation that we defined in the previous section for general graphs. When we restrict to claw-free graphs, this operation preserves the structure of the graph.

**Lemma 11.** *Let  $G$  be a claw-free graph with stability number  $\alpha(G) \geq 4$ . Then  $G$  contains a 5-wheel.*

The second lemma shows that a connected claw-free  $G$ , that is not quasi-line and has stability number at least four, is indeed the gluing of two claw-free strips.

**Lemma 12.** *Let  $G(V, E)$  be a claw-free graph with  $\alpha(G) \geq 4$ . Let  $W := (a; u_1, u_2, u_3, u_4, u_5)$  be a 5-wheel in  $G$ . Let  $(H_1, a_1, b_1)$  and  $(H_2, a_2, b_2)$  be two claw-free strips with  $a_1 = a_2 = a$  and  $b_1 = b_2 = b$ . Let  $G' = H_1 \cup H_2$  be the gluing of  $H_1$  and  $H_2$  at  $(a, b)$ . Then  $\alpha(G'[H_1 \setminus \{a_1, b_1\}]) \leq 3$  and  $(a_1, b_1) \notin E(H_1)$ . Moreover,  $G'$  is claw-free and  $|V(G')| = O(n^3)$ .*

**Lemma 13.** *Let  $G(V, E, w)$  be a claw-free graph with  $\alpha(G) \geq 4$ . Let  $n = |V(G)|$  and  $w : V \mapsto \mathbb{R}$  be a weight function. Let  $\tilde{G}(\tilde{V}, \tilde{E}, \tilde{w})$  be a graph with  $n$  vertices and  $\tilde{w} : \tilde{V} \mapsto \mathbb{R}$  a weight function.*

- $\alpha_w(G) = \alpha_{\tilde{w}}(\tilde{G})$  and  $|\tilde{V}| \leq n$ .
- $\alpha(\tilde{G}) \leq 3$ .

*Moreover,  $\tilde{G}$  can be constructed in time  $O(n^4)$  from  $G$  and  $w$ . The construction of  $\tilde{G}$  is in  $O(n)$ .*

The statement is trivial if either  $G$  is quasi-line or  $\alpha(G) \leq 3$ , so suppose  $G$  is not quasi-line and with  $\alpha(G) \geq 4$ . There must exist a vertex  $v$  with an odd-hole in  $\overline{G[N(v)]}$ . Indeed detecting an odd-hole in a graph  $H$  that is triangle-free is standard ( $\overline{G[N(v)]}$  is triangle-free since  $G$  is claw-free) and can be done in  $O(m)$ . We need to visit an auxiliary graph  $H'$ , where each vertex  $u \in V(H)$  is duplicated into  $u', u''$  and, for each edge  $(u, v) \in E(H)$ , we add two edges  $(u', v'')$  and  $(u'', v')$  in  $H'$ . It is not difficult to see that a path of minimum length connecting  $u'$  to  $u''$  corresponds, after shrinking duplicated vertices, to a shortest odd hole in  $H$ . Since  $G$  is not quasi-line, in time  $O(nm)$  we must detect an odd-hole in some  $\overline{G[N(v)]}$ : we take the smallest one. This has to be a 5-hole by Lemma 10.

So we have detected a 5-wheel in  $G$ . Using Lemma 12, we build in time  $O(n^3)$  the claw-free strips  $(H_1, a_1, b_1)$  and  $(H_2, a_2, b_2)$ , where  $H_1$  has an induced subgraph that is a 5-wheel and  $(a_1, b_1) \notin E(H_1)$ . We replace  $H_1$  by a gem and define the graph  $G'(V', E', w')$  that is the gluing of  $(H_2, a_2, b_2)$  with the gem, as in Lemma 3. Therefore,  $\alpha_w(G) = \alpha_{w'}(G') + w_{\overline{AUB}}$ . Note that the crucial stable sets for  $H_1$ , and therefore the weights  $w'$  on the vertices of the gem, can be computed in  $O(n^3)$  by enumeration since  $\alpha(G'[H_1 \setminus \{a_1, b_1\}]) \leq 3$ .  $G'$  is claw-free because of Lemma 11.

Observe that, all together,  $G'$  can be built in time  $O(n^3)$  from  $G$ . Now observe that  $G'$  contains less vertices (at least two less) than  $G$ . If either  $\alpha(G') \leq 3$  or  $G'$  is quasi-line, we stop and let  $\tilde{G} = G'$ , else we repeat this procedure (at most  $O(n)$  times since we remove at least two vertices each time). Therefore,  $\tilde{G}$  can be built in time  $O(n^4)$  from  $G$  and has less than  $n$  vertices. Finally, we may build an MWSS of  $G$  from an MWSS of  $\tilde{G}$  in time  $O(n)$  by applying inductively Remark 4. The statement follows. □

We are now left to solve the stable set problem in quasi-line graphs. Even if quasi-line graphs are more friendly from a structural point of view than claw-free graphs, there is a configuration that always annoyed us and can be easily removed when dealing with the stable set problem:

### 4 From Quasi-Line Graphs to Bipolar-Free Quasi-Line Graphs

Lovász and Plummer [6] called a vertex of a claw-free graph such that  $\{v\} \cup N(v)$  can be covered by two maximal cliques. We say that  $v$  is when this covering is unique. Recall that  $U(v)$  is the set of vertices that are universal to some vertex  $v$ .

**Lemma 14.**  $\frac{v}{G(V, E)} (H_1 \cup U(v) \cup \{v\}, H_2 \cup U(v) \cup \{v\}) (H_1, H_2) V(H)$

**Definition 15.**  $G (v_1, v_2) K (N(v_1) \cup N(v_2)) \setminus K$

A quasi-line graph without bipolar pairs of vertices is said to be . The notion of bipolar pair of vertices for quasi-line graphs is closely related to the more general concept of semi-homogeneous pair of cliques that we defined in Section 2.2, as it is confirmed by the following lemma.

**Lemma 16.**  $G (v_1, v_2) A = \{v_1, v_2\} B = (N(v_1) \setminus (N(v_2) \cup \{v_2\})) \cup (N(v_2) \setminus (N(v_1) \cup \{v_1\})) (A, B) G$

We have to prove that a vertex  $v \notin A \cup B$  that is neither complete to  $A$  nor to  $B$  is anticomplete to  $A \cup B$ . By hypothesis,  $(v_1, v_2)$  is a bipolar pair, i.e. there exists a maximal clique  $K$  such that  $v_1, v_2 \in K$  and  $(N(v_1) \cup N(v_2)) \setminus K$  is a clique. Assume w.l.o.g. that  $v$  is not adjacent to  $v_2$ . In particular,  $v \notin K$ . Moreover,  $v$  is not adjacent to  $v_1$ , since otherwise,  $v \in (N(v_1) \setminus N(v_2)) \subseteq B$ . Assume that  $v$  is not anti-complete to  $B$ , then there exists  $w$  w.l.o.g. in  $N(v_2) \setminus N(v_1)$  that is adjacent to  $v$  ( $w$  exists since  $v_1$  is not universal to  $v_2$ ). But then  $v$  is complete to  $N(v_1) \setminus N(v_2)$ . Indeed suppose that there is  $z \in N(v_1) \setminus N(v_2)$  not adjacent to  $v$ , then  $(w, z, v_2, v)$  is a claw. But we can apply the same reasoning with  $w \in N(v_1) \setminus N(v_2)$  to show that  $v$  is complete to  $N(v_2) \setminus N(v_1)$ . Thus  $v$  is complete to  $B$  a contradiction.  $\square$

**Lemma 17.**  $G(V, E, w) n, w : V \mapsto \mathbb{R} G'(V, E', w) E' \supseteq E \alpha_w(G) = \alpha_w(G') G' O(n^6)$

Let  $w : V \mapsto \mathbb{R}$  be the weight function on the vertices of  $G$ . Our procedure goes as follows. We detect a bipolar pair of vertices for  $G$ , if any. If  $G$  has no bipolar pairs of vertices, we stop. Else, let  $v_1, v_2$  be a bipolar pair for  $G$ . We know from Lemma 16 that  $A = \{v_1, v_2\}$  and  $B = (N(v_1) \setminus (N(v_2) \cup \{v_2\})) \cup (N(v_2) \setminus (N(v_1) \cup \{v_1\}))$  form a semi-homogeneous pair of clique. Without loss of generality assume that  $\{v_2, w_1\}$  is an MWSS of size two in  $A \cup B$ . We define a graph  $G'$  by adding to  $G$  every edge different from  $(v_2, w_1)$  between a vertex of  $A$  and vertices of  $B$ . We know from Lemma 9 that  $\alpha_w(G) = \alpha_w(G')$  and every stable set of  $G'$  is a stable set of  $G$  with the same weight. The graph  $G'$  is still quasi-line (see the following). If  $G'$  has no more bipolar pairs of vertices, we stop; else we iterate. Since we cannot add more than  $O(n^2)$  edges to  $G$ , we iterate at most  $O(n^2)$  times.

Our proof follows by induction if we prove that:

- 1 Detecting a bipolar pair of vertices of a quasi-line graph  $G$ , if any, can be done in time  $O(n^4)$ .
- 2  $G'$  is still quasi-line.

1. Let  $u$  and  $v$  be a pair of vertices that are not universal to each other. It is straightforward to see that they form a bipolar pair if and only if  $\overline{G[N(u) \cup N(v)]}$  is bipartite and admits a bi-coloring where  $u$  and  $v$  get the same color. Therefore, we can detect a bipolar pair of vertices in time  $O(n^4)$ .

2. We consider the case where  $G' = G \cup (v_1, w_2)$ , i.e.  $A$  is complete to  $B$  but for the edges  $\{v_2, w_1\}$  and  $(v_1, w_2)$  (the general case follows easily by induction).

We want to show that  $G'$  is still quasi-line, i.e. the neighborhood of each vertex can be covered by two cliques. Observe that, by definition,  $E(G) \subset E(G')$  and, by hypothesis,  $G$  is quasi-line. Therefore, for any vertex  $v \notin \{v_1, w_2\}$ ,  $N_{G'}(v)$  can be still covered by two cliques, since  $N_{G'}(v) = N_G(v)$ . As far as  $v_1$  and  $w_2$  are concerned, it will be enough to show that:

- (i) there exist two cliques of  $G$  covering  $N_G(v_1) \cup \{w_2\}$ ;
- (ii) there exist two cliques of  $G$  covering  $N_G(w_2) \cup \{v_1\}$ .

First, we need some definitions and a claim. Let  $K_1$  ( $K_2$ , resp.) be a maximal clique such that  $K$  and  $K_1$  ( $K$  and  $K_2$ , resp.) cover  $N(v_1)$  ( $N(v_2)$ , resp.).

Denote by  $S_1, S_2, \dots, S_8$  a partition of the vertex set  $V(G)$  of  $G$ , defined as follows:  $S_1 = \{v : v \in (K \cap K_1) \setminus K_2\}$ ,  $S_2 = \{v : v \in (K \cap K_2) \setminus K_1\}$ ,  $S_3 = \{v : v \in K_2 \setminus (K_1 \cup K)\}$ ,  $S_4 = \{v : v \in K_1 \setminus (K_2 \cup K)\}$ ,  $S_5 = \{v : v \in K \cap K_1 \cap K_2\}$ ,  $S_6 = \{v : v \in (K_1 \cap K_2) \setminus K\}$ ,  $S_7 = \{v : v \in K \setminus (K_1 \cup K_2)\}$ ,  $S_8 = \{v : v \in V \setminus (K \cup K_1 \cup K_2)\}$ .

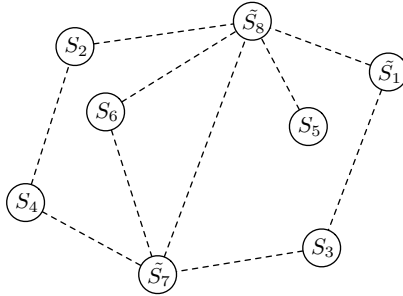
**Claim 18.**  $\dots, z \in S_8, (z, w_2) \in E, (z, w_1) \in E, z \in S_3 \cup S_4$

W.l.o.g. suppose  $\exists z \in S_8$  s.t.  $(z, w_2) \in E$ . First we prove that  $z$  is complete to  $S_4$ . In fact,  $S_4, z, v_2 \subseteq N(w_2)$ . Moreover  $S_4 \cup \{z\}$  is anticomplete to  $v_2$ . Thus,  $S_4 \cup \{z\}$  must be a clique, since  $G$  is quasi-line. Now we prove that  $z$

is complete to  $S_3$ . In fact,  $S_3, z, v_1 \subseteq N(w_1)$ . Moreover  $S_3$  is not complete to  $v_1$  and  $z$  is not adjacent to  $v_1$ . Thus,  $S_3 \cup \{z\}$  must be a clique, since  $G$  is quasi-line.

(i). From the above definitions,  $N_G(v_1) = S_1 \cup S_5 \cup S_2 \cup S_7 \cup S_4 \cup S_6$ . Since  $S_1 \cup S_5 \cup S_2 \cup S_7 = K$  and  $S_4 \cup S_6 \cup \{w_2\}$  are both cliques of  $G$ , by hypothesis or construction, the statement follows.

(ii). We have:  $N_G(w_2) = S_3 \cup S_6 \cup S_4 \cup S_2 \cup S_5 \cup \tilde{S}_1 \cup \tilde{S}_7 \cup \tilde{S}_8$ , where  $\tilde{S}_i$  denotes the set of vertices belonging to  $S_i$  that are adjacent to  $w_2$ . Observe that  $v_1$  is complete to  $S_6 \cup S_2 \cup S_5 \cup \tilde{S}_1 \cup \tilde{S}_7$ . Therefore, in order to prove our statement, it is enough to show that there exist two cliques  $K_L, K_R$  of  $G$  covering  $N_G(w_2)$  and such that  $S_3 \cup S_4 \cup \tilde{S}_8$  belongs to a same clique, say  $K_L$ . This is the same as showing that there is a valid bi-coloring of the bipartite graph  $H = G[N_G(w_2)]$  such that  $S_3, S_4, \tilde{S}_8$  get the same color, which for our purposes will be either red or blue. In the following, we build such a coloring.



**Fig. 1.** The possible adjacencies in  $H$ . Two sets are connected by a dotted edge if and only if they are not complete to each other in  $G$ .  $S_4$  and  $S_3$  are complete by hypothesis;  $\tilde{S}_8$  is complete to  $S_3 \cup S_4$  by Claim 13; other pairs that are complete belong to a same clique in  $\{K, K_1, K_2\}$ .

The graph  $H$  is in general not connected and in Fig. 1 we represent the possible adjacencies in  $H$  among the sets  $\tilde{S}_1, S_2, S_3, S_4, S_5, S_6, \tilde{S}_7, \tilde{S}_8$ , i.e. two sets are connected by a dotted edge if and only if they are not complete to each other in  $G$ . Consider the component  $C_1$  in  $H$  containing  $v_2$  and say  $v_2$  is blue. The set  $S_4$  and  $\tilde{S}_8$  are anticomplete in  $G$  to  $v_2$  and thus they are in  $C_1$  and they have to be red. By maximality of the clique  $K_1$ , each vertex in  $S_2$  has a non-neighbor in  $S_4$  and thus  $S_2$  is in  $C_1$  and has to be blue. Consider a vertex  $v \in S_5$ . By definition,  $v$  is complete to each  $S_i$  but, possibly,  $S_8$ . Therefore, if  $v$  is complete to  $\tilde{S}_8$ , then it is a singleton for  $H$  and thus can be made arbitrarily blue; else  $v$  belongs to  $C_1$  and it is again blue.

So far we have seen that there exists a valid partial bi-coloring of  $H$  such that w.l.o.g.  $S_4$  and  $\tilde{S}_8$  are red while  $S_2$  and  $S_5$  are blue. We now show that also the vertices of  $S_3$  that are in  $C_1$  are red. Observe that, since  $H$  is bipartite and  $C_1$

is connected, the coloring of the vertices of  $C_1$  is forced: vertices that have even distance from the vertices in  $S_4 \cup \tilde{S}_8$  are red, vertices that have odd distance from the vertices in  $S_4 \cup \tilde{S}_8$  are blue; therefore it is enough to show that no vertex of  $S_3 \cap V(C_1)$  has odd distance from some vertex in  $S_4 \cup \tilde{S}_8$ . Therefore, suppose to the contrary that  $s_3 \in S_3$  and  $x \in S_4 \cup \tilde{S}_8$  have odd distance and assume w.l.o.g. their distance to be minimum. Let  $P = (s_3, u_2, u_3, \dots, u_{2k} \equiv x)$  be a path attaining such minimum distance.

We base our analysis on the possible edges in  $H$  from the graph of Fig. [11](#). Recall that the sets  $\tilde{S}_1, S_2, S_3, S_4, S_5, S_6, \tilde{S}_7$  are cliques for  $G$  (and therefore stable sets for  $H$ ). It is clear that  $P$  does not contain any vertex of  $\tilde{S}_1$  (since  $P$  is of minimum distance between  $S_3$  and  $S_4 \cup \tilde{S}_8$ , in this case it would be a path of length 2 from  $S_3$  to  $\tilde{S}_8$ ). Analogously,  $P$  cannot take any vertex from  $S_4$ , else  $P$  would be an even path from  $s_3$  to  $S_4$ , and it cannot take any edge from  $\tilde{S}_7$  to  $S_8$ , else  $P$  would be an even path from  $S_3$  to  $\tilde{S}_8$ . It is thus clear that  $P$  alternates between vertices of  $\tilde{S}_7$  and  $S_6$  before ending with a vertex of  $S_8$ : namely,  $u_3, u_5, \dots, u_{2k-1} \in S_6$  and  $u_2, u_4, \dots, u_{2k-2} \in \tilde{S}_7$ . Let  $V(P)$  be the vertices in  $P$ ; it follows that the vertices in  $V(P) \cap S_6$  are blue while the vertices in  $V(P) \cap \tilde{S}_7$  are red. Therefore, in  $H$  there cannot be adjacencies between vertices in  $S_4$  and vertices in  $V(P) \cap \tilde{S}_7$ , that is, vertices in  $S_4$  are complete to vertices in  $V(P) \cap \tilde{S}_7$  in  $G$ . Now observe that  $P$  is of minimum distance, hence it is an induced odd path of  $H$ . The subgraph of  $H$  induced by  $V(P) \cup \{v_1\}$  is an induced odd-hole ( $v_1$  is complete in  $G$  to  $S_6, \tilde{S}_7$  and anticomplete to  $\tilde{S}_8$  and  $S_3$ ). Therefore  $G[V(P) \cup \{v_1\}]$  is an induced anti-hole. But  $(V(P) \cup \{v_1\}) \subseteq N_G(w_1)$ . Indeed  $w_1$  is complete to  $S_6, v_1, S_3$  by definition, to  $\tilde{S}_8$  by the previous claim and, since it belongs to  $S_4$ , it is complete to  $V(P) \cap \tilde{S}_7$  as we have just observed. It follows that  $N_G(w_1)$  contains an odd antihole, but this is in contradiction with  $G$  being quasi-line. Therefore, the vertices of  $S_3$  that are in  $C_1$  are red.

Finally, let  $Q$  be the set of vertices of  $H$  that are not in  $C_1$  and are not from  $S_5$ . It follows from above that  $Q \subseteq \tilde{S}_1 \cup \tilde{S}_7 \cup S_3 \cup S_6$ . Clearly, any coloring for the vertices of  $Q$  that is valid for  $H[Q]$  is also valid for  $H$ . In particular, we may give color red to the vertices of  $Q \cap (S_3 \cup S_6)$  and color blue to the vertices in  $Q \cap (\tilde{S}_1 \cup \tilde{S}_7)$ .

We have therefore built a valid bi-coloring for  $H$  where  $S_3, S_4, \tilde{S}_8$  get the same color (red) and statement (ii) is proved.  $\square$

From Lemma [13](#) and Lemma [17](#), we know that in order to solve the MWSS problem in claw-free graphs, we only need to be able to solve the MWSS problem in bipolar-free quasi-line graphs. We will now analyze the structure of those graphs in order to devise a polynomial time algorithm.

## 5 A Decomposition Theorem for Bipolar-Free Quasi-Line Graphs

In this section we give our main structural result, concerning the structure of bipolar-free quasi-line graphs. In particular, we will show that a rich class of

bipolar-free quasi-line graphs is the composition of suitable strips that can be found by identifying articulation cliques.

**Definition 19.** Let  $K$  be a net clique of  $G$ . An articulation clique  $K$  is a net clique  $K$  such that for each  $v \in K$ ,  $N(v) \setminus K$  is a claw.

An articulation clique  $K$  of a quasi-line graph  $G(V, E)$  is called strongly regular if each vertex  $v \in K$  is strongly regular. Observe that  $K$  is a strongly regular articulation clique if and only if, for each  $v \in K$ , there is a unique pair of maximal cliques covering  $N(v) \cup \{v\}$  and  $K$  is one of these cliques. Detecting if a quasi-line graph has a strongly regular articulation clique is therefore easy.

**Lemma 20.** Let  $G$  be a quasi-line graph with  $n$  vertices. Then the number of strongly regular articulation cliques in  $G$  is  $O(n^3)$ .

In order to detect a strongly regular articulation clique, we first build the set  $R$  of strongly regular vertices and, for each vertex  $v \in R$ , the unique pair of maximal cliques  $(K_1(v), K_2(v))$  covering  $N(v) \cup \{v\}$ . That can be done in time  $O(n^3)$ , thanks to Lemma 14. Let  $K(R) = \{K_1(v), K_2(v), v \in R\}$ . A clique  $K$  of  $G$  is a strongly regular articulation one if and only if  $K \subseteq R$  and, for each  $v \in K$ ,  $K \in \{K_1(v), K_2(v)\}$ . Since  $|K(R)| \leq 2n$ , it follows that we can list all strongly regular articulation cliques in time  $O(n^3)$ .  $\square$

**Definition 21.** Let  $K$  be a net clique of  $G$ . A net clique  $K$  is called a claw-free net clique if there exists a set  $\{s_1, s_2, s_3\} \subseteq N(K)$  such that  $K$  is a claw-free net clique with respect to  $\{s_1, s_2, s_3\}$ .

**Lemma 22.** Let  $K$  be a net clique of  $G$ . Then  $K$  is a claw-free net clique if and only if it is a strongly regular articulation clique.

It is enough to show that, for every vertex  $v \in K$ , there exists a maximal clique  $K(v)$  such that  $(K, K(v))$  is the unique covering of  $N(v)$  into two maximal cliques.

By definition, there exists a stable set  $\{s_1, s_2, s_3\} \subseteq N(K)$  and each  $v \in K$  is adjacent to at most one vertex in  $\{s_1, s_2, s_3\}$ . So let  $K_1 = K \cap N(s_1)$ ,  $K_2 = K \cap N(s_2)$ ,  $K_3 = K \cap N(s_3)$ ,  $K_4 = K \setminus (K_1 \cup K_2 \cup K_3)$  ( $K_1, K_2, K_3 \neq \emptyset$  since  $\{s_1, s_2, s_3\} \subseteq N(K)$ ).

First, suppose  $v \in K_1$ . Let  $(Q_1, Q_2)$  be a pair of maximal cliques such that  $N(v) \cup \{v\} = Q_1 \cup Q_2$  (such a pair exists, since the graph is quasi-line). Assume w.l.o.g. that  $s_1 \in Q_1$ , it follows that  $K \setminus K_1 \subseteq Q_2$ . We now show that every vertex  $z \in N(v) \setminus K$  is not complete to  $K \setminus K_1$ . Suppose the contrary, i.e. there exists  $z \in N(v) \setminus K$  that is complete to  $K \setminus K_1$ . Since  $K$  is maximal, there exists  $w \in K_1$ ,  $w \neq v$ , such that  $(w, z) \notin E$ . Since  $z$  is adjacent to  $v$ , it cannot be adjacent to both  $s_2$  and  $s_3$  (otherwise there would be the claw  $(z; s_2, s_3, v)$ ). Assume w.l.o.g.  $z$  is not linked to  $s_3$ . Let  $z_3$  be a vertex in  $K_3$ . Then  $(z_3; s_3, w, z)$



is a claw, a contradiction. Therefore, every vertex in  $z \in N(v) \setminus K$  is not complete to  $K \setminus K_1$  and so it must belong to  $Q_1$ . It follows that  $Q_1 = (N(v) \setminus K) \cup \{v\} \cup U(v)$  and  $Q_2 = K$ , that is,  $(Q_1, K)$  is the unique covering of  $N(v)$  into two maximal cliques. The same holds for any vertex  $v$  in  $K_2$  or  $K_3$ .

Now suppose that  $v \in K_4$ . If  $v$  is a simplicial vertex, then the statement is trivial. Now suppose that there exists  $w \notin K$  such that  $(w, v) \in E$ . Observe that  $w$  is adjacent to at most one vertex of  $\{s_1, s_2, s_3\}$ : if the contrary, assume w.l.o.g.  $s_1, s_2 \in N(w)$ , there would be the claw  $(w; v, s_1, s_2)$ . Hence there exists a stable set of size three in  $\{w, s_1, s_2, s_3\}$  containing  $w$  and we are back to the previous case.  $\square$

**Definition 23.** Let  $K$  be an articulation clique in  $G$ . For  $v \in K$ , define

$$Q(v) = \begin{cases} \{v\} & \text{if } v \text{ is simplicial} \\ Q(v) = U(v) \cup \{v\} & \text{otherwise} \end{cases}$$

Let  $\mathcal{Q}(K) = \{Q(v) \mid v \in K\}$ . We say that  $v$  generates  $Q$  if  $Q \in \mathcal{Q}(K)$  and  $Q = Q(v)$ .

**Lemma 24.** Let  $G(V, E)$  be a graph and  $K \subseteq V$  be an articulation clique. Then

- (i)  $\forall v \in K, v \in Q(v) \subseteq K$
- (ii)  $\forall u, v \in K, u \in Q(v) \implies Q(u) \subseteq Q(v)$
- (iii)  $G \setminus K = \bigcup_{Q \in \mathcal{Q}(K)} Q$
- (iv)  $G \setminus K = \bigcup_{Q \in \mathcal{Q}(K)} N(Q) \setminus K$

Let  $v$  be a vertex of  $K$ . (i) It is trivial if  $v$  is simplicial. Else, let  $u$  be a vertex in  $N(v) \setminus K$ . Since  $K$  is maximal, there exists some vertex  $z \in K$  such that  $(u, z) \notin E$ ; therefore  $u$  is not universal to  $v$  and does not belong to  $Q(v)$ . (ii) Suppose that  $u, v \in V, u \neq v$ . Observe that  $v$  is not simplicial. The statement is trivial if  $u$  is simplicial. So assume that it is not. Let  $z \in Q(u)$ , we want to show that  $z \in Q(v)$ . It is trivial if  $z \equiv u$ , so assume that  $z \neq u$ . By definition,  $N(u) \cup \{u\} \subseteq N(z) \cup \{z\}$ . On the other hand, since  $u \in Q(v)$ ,  $N(v) \cup \{v\} \subseteq N(u) \cup \{u\}$ . Therefore,  $N(v) \cup \{v\} \subseteq N(z) \cup \{z\}$ . (iii) It follows from (i) that, in order to show that  $\mathcal{Q}(K)$  defines a partition of  $K$ , it is enough to show that the family  $\{Q(v), v \in K\}$  is laminar. Suppose to the contrary that there exist  $u, v$  with  $Q(v) \cap Q(u), Q(v) \setminus Q(u), Q(u) \setminus Q(v) \neq \emptyset$ . Therefore  $u$  and  $v$  are not simplicial and it follows from (ii) that  $v \notin Q(u)$  and  $u \notin Q(v)$ . That is,  $u$  and  $v$  are not universal to each other. Therefore, there exists  $z$  such that  $(z, v) \in E$  and  $(z, u) \notin E$ . Similarly, there exists  $y$  such that  $(y, u) \in E$  and  $(y, v) \notin E$ . Let  $w \in Q(v) \cap Q(u)$ . Observe that  $N(u) \setminus K$  and  $N(v) \setminus K$  belong to  $N(w)$ , since  $w$  is universal to  $u$  and  $v$ . On the other hand,  $K$  is an articulation clique, therefore  $N(w) \setminus K$  is a clique. It follows that  $(N(u) \cup N(v)) \setminus K$  is a clique. Then  $(u, v)$  is a bipolar pair, a contradiction. (iv) Suppose the contrary. There exist  $x, y \in N(Q) \setminus K$  such that  $(x, y) \notin E$ . Since  $K$  is an articulation

clique, it follows that no vertex of  $Q$  is joined to both  $x$  and  $y$ . Therefore there exist  $u, v \in Q$  such that  $(u, y) \in E$ ,  $(u, x) \notin E$ ,  $(v, x) \in E$ ,  $(v, y) \notin E$ . Neither  $u$  nor  $v$  can generate  $Q$ , as  $(u, y) \in E$  and  $(u, x) \notin E$ , as well as  $(v, x) \in E$  and  $(v, y) \notin E$ . Therefore there exists  $w \in K$  that generates  $Q$ . Such a vertex  $w$  is not simplicial (otherwise it could not generate  $Q$ ), thus there exists a vertex  $n \notin K$  such that  $(n, w) \in E$ . The neighbors of  $w$  outside  $K$  must be universal to  $u$  and  $v$ , thus  $(n, u) \in E$  and  $(n, v) \in E$  (therefore  $n \notin \{x, y\}$ ). For the same reason  $x, y \notin N(w)$ . Finally,  $(n, x) \in E$  and  $(n, y) \in E$ , otherwise  $u$  or  $v$  would be vertices of  $K$  with a stable set of size 2 in their neighborhood outside  $K$ . It follows that  $(n; x, y, w)$  is a claw, which is a contradiction.  $\square$

**Definition 25.** Let  $G$  be a graph and  $K$  a subset of vertices. The ungluing operation  $\mathcal{A}(K)$  on  $G$  is defined as follows:  $\mathcal{A}(K) := \{a_1, \dots, a_k\}$  where  $a_i \in N_{G|K}(a_i) = Q_i$  for  $i \in \{1, \dots, k\}$  and  $i \neq j$ .

We defer to the journal version of this paper the proof of the following lemma, showing that the ungluing operation preserves quasi-lineness, bipolar-freeness and has other useful properties.

**Lemma 26.** Let  $K$  be a subset of vertices of  $G$ . Then:

- (i)  $G|K$  is a graph.
- (ii)  $G|K$  is a graph.
- (iii)  $\mathcal{A}(K)$  is a graph.
- (iv)  $w \notin K$  is a vertex of  $G$  such that  $w \in N_{G|K}(w)$ .

### 5.1 Distance Simplicial Graphs

Pulleyblank and Shepherd [10] showed that, given a fixed  $k$ , the MWSS problem can be solved via longest paths in an acyclic digraph in time  $O(n^{k+1})$  for connected graphs with a vertex  $v$  having  $\alpha(N_j(v)) \leq k$  for all  $j$ . This motivated them to define a connected graph such that, for every  $v$  and every  $j$ ,  $\alpha(N_j(v)) \leq 2$ . Trivially, distance claw-free graphs are a subclass of claw-free graphs and it follows from what above that one can solve the MWSS problem in distance claw-free graphs in time  $O(n^3)$ . They also proved that a connected claw-free graph that is not distance claw-free has an induced net. We have therefore the following lemma, whose proof is omitted.

**Lemma 27.** Let  $G$  be a graph and  $G$  a graph.

**Definition 28.** Let  $G(V, E)$  be a distance simplicial graph and let  $v \in V$ . For  $j \geq 1$ , let  $N_j(v)$  denote the set of vertices at distance  $j$  from  $v$ .

It follows from above that the MWSS problem can be solved in time  $O(n^2)$  for distance simplicial graphs. It is possible to show that a distance simplicial graph is distance claw-free, but, since we do not need this statement for the following, we defer it and its proof to the journal version of this paper.

We also defer the proof of the following lemma, showing that a claw-free graph with a strongly simplicial vertex is either distance simplicial or has a net-clique  $K$ , that we can use to unglue  $G$ . Moreover,  $G|_K$  has some useful properties.

**Lemma 29.** Let  $G(V, E)$  be a claw-free graph with a strongly simplicial vertex  $a$ . Let  $K$  be a net-clique in  $G$  such that  $K = N_{j-1}(a) \cup H$  for some  $H \subseteq N_{j-2}(a)$  and  $j \geq 3$ .

- (i)  $|K| \leq O(n^2)$
- (ii)  $K$  is a net-clique in  $G|_K$ .
- (iii)  $G|_K$  is distance simplicial.
- (iv) Let  $a'$  be a vertex in  $N_{j-2}(a)$  such that  $a' \equiv a$ . Then  $G|_K$  is distance simplicial with respect to  $a'$ .

### 5.2 The Decomposition Algorithm

We are now ready to define our decomposition procedure (cf. Algorithm 1). The algorithm receives a connected graph  $G$  that is quasi-line and bipolar-free, but not distance claw-free. It returns: a graph  $G^L$  (still quasi-line and bipolar-free) such that  $V(G^L) = V(G) \cup A^L$  (the vertices in  $A^L$  are artificial) and such that each component is distance simplicial; a partition  $\mathcal{P}$  of the vertices in  $A^L$ . As we show later, the components of  $G^L$  and the partition  $\mathcal{P}$  can be used to define a strip decomposition of  $G$ .

---

**Algorithm 1.** An algorithm to decompose a bipolar-free quasi-line graph that is not distance claw-free

---

**Require:** A quasi-line and bipolar-free connected graph  $G$  that is not distance claw-free.

**Ensure:** A graph  $G^L$  such that  $V(G^L) = V(G) \cup A^L$ ; a partition  $\mathcal{P}$  of the vertices in  $A^L$ .

- 1: Detect a strongly regular articulation clique  $K$  and unglue it.
  - 2: Let  $G^0 := G|_K$ . Let  $l = 0$ ,  $A^0 := \mathcal{A}(K)$  and  $\mathcal{P} := \{\mathcal{A}(K)\}$ .
  - 3: **while**  $\exists a \in A^l$  such that  $\alpha(N_j(a)) > 1$  for some  $j \geq 3$  **do**
  - 4:   Let  $K^l$  with  $K^l \cap A^l = \emptyset$  be a net clique.
  - 5:   Unglue  $K^l$  i.e. define  $G^{l+1} := G|_{K^l}$ ,  $\mathcal{P} := \mathcal{P} \cup \{\mathcal{A}(K^l)\}$  and  $A^{l+1} := A^l \cup \mathcal{A}(K^l)$ .
  - 6:   Let  $l := l + 1$ .
  - 7: **end while**
  - 8: Let  $L = l$ .
-

In the algorithm, we start with a connected graph  $G$  that is quasi-line and bipolar-free but not distance claw-free. Hence  $G$  has a net clique by Lemma 27 and therefore a strongly regular articulation clique  $K$  by Lemma 22. We unglue  $K$  and set  $A^0$  equal to the set of artificial vertices  $\mathcal{A}(K)$ . By statements (i) and (ii) of Lemma 26,  $G^0 := G|_K$  is quasi-line and bipolar-free, moreover, by statement (iii), the vertices in  $A^0$  are pairwise non-adjacent, every vertex of  $A^0$  is strongly simplicial and in every component of  $G^0$  there is a vertex from  $A^0$  and a vertex from  $G$ .

We then check whether in some component of  $G^0$  there exists a vertex  $a \in A^0$  such that  $\alpha(N_j(a)) > 1$  for some  $j$ . If not, by Lemma 29, each component of  $G^0$  is distance simplicial and the algorithm terminates; else, there exists a net clique  $K^0$ , that we unglue, setting  $G^1 := G^0|_{K^0}$  and  $A^1 := A^0 \cup \mathcal{A}(K^0)$ . Observe that, by Lemma 29, no vertex of  $K^0$  is strongly simplicial, therefore  $K^0 \cap A^0 = \emptyset$ ,  $G^1$  has at least one component more than  $G^0$  and each vertex of  $A^0$  is strongly simplicial in  $G^1$  (we can use the fourth statement of Lemma 29 since the vertices in  $A^0$  are pairwise non-adjacent).

By Lemma 26,  $G^1$  is quasi-line, bipolar-free, the vertices in  $\mathcal{A}(K^0)$  are pairwise non-adjacent and every vertex of  $\mathcal{A}(K^0)$  is strongly simplicial in  $G_1$ . Also in every component of  $G_1$  there is a vertex of  $A^1$  and a vertex from  $G$  (about this last statement, observe that each new component has a vertex from  $K^0$  and  $K^0 \cap A^0 = \emptyset$ ). Moreover the vertices in  $A^1$  are still pairwise non-adjacent: we have already seen that vertices in  $A^0$  are pairwise non-adjacent and vertices in  $\mathcal{A}(K^0)$  are pairwise non-adjacent. Now observe that  $A^0$  is anti-complete to  $\mathcal{A}(K^0)$ : this is because the only adjacencies defined during the reduction of  $K^0$  are adjacencies between vertices of  $\mathcal{A}(K^0)$  and vertices of  $K^0$  and no vertex of  $A^0$  are adjacent to  $K^0$ .

We set  $l = 1$  and iterate. It follows, by induction, that at each step  $l$  of the algorithm: the graph  $G^l$  is quasi-line and bipolar-free; the set  $A^l$  is made of pairwise non-adjacent strongly simplicial vertices; in each component of  $G^l$  there is at least one vertex from  $A^l$  and at least one vertex from  $G$ . Therefore, at each step, unless each component of  $G^l$  is distance simplicial, there is a net clique to unglue. The algorithm will terminate after  $L$  iterations when no component of  $G^L$  has a vertex  $a \in A^L$  such that  $\alpha(N_j(a)) > 1$  for some  $j$ , i.e. when each component of  $G^L$  is distance simplicial. In particular, since we cannot have more than  $n = |V(G)|$  components with at least a vertex from  $G$ , it follows that  $L \leq n$ . Moreover, since each iteration can be done in time  $O(n^3)$ , the running time of the algorithm is  $O(n^4)$ .

We finally show how the components of  $G^L$  and  $\mathcal{P}$  can be used to define a strip decomposition of  $G$ . Observe that, by construction,  $V(G^L) = V(G) \cup A(L)$  and  $\mathcal{P}$  defines a partition of the vertices in  $A^L$ .

**Lemma 30.**  $G^L$   $A^L$

Suppose to the contrary that there exists a connected component  $C$  in  $G^L$  with more than two vertices from  $A^L$ . Let  $a_1, a_2, a_3 \in C \cap A^L$ . In the following, we refer to the graph  $G^L$ . Recall that each vertex in  $A^L$  is strongly simplicial

and that the vertices in  $A^L$  are pairwise non-adjacent. Assume  $a_2 \in N_j(a_1)$  and  $a_3 \in N_k(a_1)$  for some  $k \geq j \geq 1$ . Actually,  $j > 1$  since otherwise  $a_1$  and  $a_2$  would be adjacent. Since each component of  $G^L$  is distance simplicial, it follows that  $\alpha(N_l(a_1)) \leq 1$  for all  $l \geq 1$ . Therefore,  $k > j$  and thus  $N_{j+1}(a_1) \neq \emptyset$ . Each vertex  $v \in N_{j+1}(a_1)$  is not adjacent to  $a_2$  (else  $a_2$ , that is adjacent to some vertex in  $N_{j-1}(a_1)$ , would not be simplicial). But then since there exists  $w \in N_{j-2}(a_1) \cap N_2(a_2)$ , it follows that  $\{v, w\}$  is a stable set of size two in  $N_2(a_2)$ , a contradiction.  $\square$

Suppose now that  $G^L$  has  $p$  components  $H_1, \dots, H_p$ . Each component  $H_i$ , for  $i = 1..p$ , has either two vertices  $a_i, b_i$  from  $A^L$  or one vertex  $a_i$  from  $A^L$ . W.l.o.g. assume that each component has two artificial vertices (else we might add a singleton to the component). Consider therefore the strips  $(H_1, a_1, b_1), \dots, (H_p, a_p, b_p)$ . It is easy to see that  $G$  is the composition of the strips  $(H_i, a_i, b_i)$ ,  $i = 1, \dots, p$  w.r.t. the partition  $\mathcal{P}$ : in fact,  $G^L, G^{L-1}, \dots, G^0$  is exactly the sequence defined in the alternative definition of strips-composed graph (see Def. 2).

We can summarize our previous results as follows:

**Lemma 31.**  $G$   $O(n^4)$   $\square$

**Theorem 32.**  $G$   $n$

We now have all the ingredients to state our main result.

**Theorem 33.** MWSS  $O(n^6)$   $G$

We check if there exists a stable set of size 4 in  $G$  by enumeration: this can be done in time  $O(n^4)$ . If not, we enumerate in time  $O(n^3)$  all stable sets of  $G$  of size 1, 2 and 3 and take the best one. Else, if the graph has stability number greater than 3, we use Lemma 13 to reduce to the MWSS problem in a quasi-line graph  $G^1$  in time  $O(n^4)$ . We now use Lemma 17 to reduce to the MWSS problem in a bipolar-free quasi-line graph  $G^2$  in time  $O(n^6)$ . We search for strongly regular articulation cliques in  $G^2$ : this can be done in time  $O(n^3)$  from Lemma 20. If there are no strongly regular articulation cliques, then we know from Lemma 22 that  $G^2$  has no net cliques and therefore (Lemma 27)  $G^2$  is distance claw-free; hence we use the algorithm of Pulleyblank and Shepherd 10 and find an MWSS in time  $O(n^3)$ . Else, we get a strip decomposition of  $G^2$  into distance simplicial strips from Lemma 31 in time  $O(n^4)$ . The crucial stable sets can be found in each strip in time  $O(n^2)$  (first recall that in a distance simplicial graph an MWSS can be found in time  $O(n^2)$ , then observe that each crucial stable set is an MWSS, if we give weight 0 to some suitable set of vertices). Therefore, from Lemma 5 we can find an MWSS in  $G^2$  in time  $O(n^3)$ . All together, we can solve the MWSS problem for  $G$  in time  $O(n^6)$ .  $\square$

## Acknowledgment

Defining graph reductions that can deal with the MWSS problem requires a deep understanding of the structure of claw-free graphs. Recently, Chudnovsky and Seymour (see the survey paper [3]) tackled this problem and they provided decomposition theorems for claw-free and quasi-line graphs. Though we started considering graph reductions only after we heard about this tremendous piece of work, we could finally avoid to use their whole machinery thanks to our simpler decomposition result we could prove for bipolar-free quasi-line graphs. We believe nevertheless that we would not have thought about our simpler decomposition without the decomposition theorems of Chudnovsky and Seymour.

## References

1. Berge, C.: Two theorems in graph theory. Proc. Nat. Acad. Sci. U.S.A. 43, 842–844 (1957)
2. Berge, C.: Graphs and hypergraphs. Dunod, Paris (1973)
3. Chudnovsky, M., Seymour, P.: The structure of claw-free graphs. In: Surveys in Combinatorics 2005. London Math. Soc. Lecture Note Series, vol. 327 (2005)
4. Cornuéjols, G., Cunningham, W.H.: Compositions for perfect graphs. Discrete Mathematics 55, 245–254 (1985)
5. Fouquet, J.: A strengthening of Ben Rebea’s lemma. Journal of Combinatorial Theory 59, 35–40 (1993)
6. Lovász, L., Plummer, M.D.: Matching theory. North Holland, Amsterdam (1986)
7. Lozin, V.V., Milanič, M.: A polynomial algorithm to find an independent set of maximum weight in a fork-free graph. In: Proceedings of SODA 2006, Miami, Florida, January 22–26 (2006)
8. Minty, G.J.: On maximal independent sets of vertices in claw-free graphs. Journal of Combinatorial Theory 28, 284–304 (1980)
9. Nakamura, D., Tamura, A.: A revision of Minty’s algorithm for finding a maximum weighted stable set of a claw-free graph. Journal of the Operations Research Society of Japan 44(2), 194–204 (2001)
10. Pulleyblank, W., Shepherd, F.: Formulations for the stable set polytope. In: Rinaldi, G., Wolsey, L.A. (eds.) Proceedings of IPCO 1993, Erice, Italy, April 19 - May 1 (1993)
11. Sbihi, N.: Algorithme de recherche d’un stable de cardinalité maximum dans un graphe sans étoile. Discrete Mathematics 29, 53–76 (1980)
12. Schrijver, A.: Combinatorial optimization. Polyhedra and efficiency. In: Algorithms and Combinatorics 24, 3 volumes. Springer, Berlin (2003)

# A Polynomial Algorithm for Weighted Abstract Flow<sup>\*</sup>

Maren Martens and S. Thomas McCormick

Sauder School of Business, University of British Columbia,  
Vancouver, BC V6T 1Z2

{maren.martens,tom.mccormick}@sauder.ubc.ca

**Abstract.** Ford and Fulkerson’s original 1956 max flow/min cut paper formulated max flow in terms of flows on paths, rather than the more familiar flows on arcs. In 1974 Hoffman pointed out that Ford and Fulkerson’s original proof was quite abstract, and applied to a wide range of flow problems. In this abstract model we have capacitated elements, and linearly ordered subsets of elements called paths. When two paths share an element (“cross”), then there must be a path that is a subset of the first path up to the cross and the second path after the cross. Hoffman’s generalization of Ford and Fulkerson’s proof showed that integral optimal primal and dual solutions still exist under this weak assumption. However, his proof is non-constructive.

Hoffman’s paper considers a sort of supermodular objective on the path flows, which allows him to include transportation problems and thus min-cost flow in his framework. We develop the first combinatorial polynomial algorithm that solves this problem, thereby also give a constructive proof of Hoffman’s theorem. Our algorithm accesses the network only through a dual feasibility oracle, and resembles the successive shortest path algorithm for ordinary min-cost flow. It uses some of the same techniques used to solve the max flow/min cut version of Hoffman’s model, plus a method to re-optimize when capacities change inside capacity scaling.

## 1 Introduction

For many years researchers have investigated which classes of linear programs have guaranteed integral optimal solutions. One such large class is the *totally dual integral (TDI)* LPs (see, e.g., Schrijver [10, Chapter 22]). TDI systems often have size exponential in the natural variables of the model. It is sometimes possible to show a separation algorithm that then implies a polynomial algorithm via the Ellipsoid Method (see, e.g., Grötschel, Lovász, and Schrijver [5]), but this is unsatisfactory as Ellipsoid-based algorithms are reputed to be slow in practice. We would prefer to have *combinatorial* (i.e., non-Ellipsoid) algorithms for these problems. Finding such combinatorial algorithms has been a big area of research in recent times.

There have been some notable successes recently in finding combinatorial algorithms for TDI problems. Two such problems are Submodular Function Minimization (SFM)

---

<sup>\*</sup> This work was partially supported by an NSERC Operating Grant, the DFG grants 444 USA 121/1/07, SK 58/4-1, and SK 58/5-3, and by the Graduate School of Production Engineering and Logistics, North Rhine-Westphalia.

and Bisubmodular Function Minimization (BSFM), see [8,9] (and a general algorithmic framework for TDI problems is in [1]). One of the papers originating the idea of TDI-ness is Hoffman [6]. His paper developed an abstract generalization of the path-variable version of Ford and Fulkerson’s celebrated *Max Flow-Min Cut Theorem* (MFMC Theorem) [3]. Hoffman’s model allows supermodular weights on paths, and so we call it *Weighted Abstract Flow (WAF)*; it includes all versions of the MFMC Theorem, as well as some weighted flow problems such as transportation problems and versions of min-cost flow. Hoffman’s proof was non-constructive. McCormick [7] found a polynomial algorithm for the unweighted “max flow” version of WAF, but left an algorithm for general WAF as an open problem.

WAF is important as it contains many more models as special cases than the version considered in [7]. Since there could be an exponential number of paths, algorithms for WAF interact with the abstract network through an oracle: we want to solve the problem despite wearing a blindfold that prevents us from “seeing” the network. Hence from the viewpoint of its many applications, the challenge of seeing how little information about a network is needed to solve such problems, as well as the viewpoint of wanting to develop algorithmic techniques for solving TDI problems, getting an algorithm for WAF is worthwhile.

The main result in the present paper is the first polynomial combinatorial algorithm for WAF, by adapting the Successive Shortest Path algorithm for min-cost flow to WAF to find integral solutions. This necessitates significantly revising and extending the “max flow” algorithm of [7] to a version that can deal with elements whose flow is restricted to be tight to their capacity.

## 1.1 Hoffman’s Model

We are given a finite set  $E$  of (capacitated) *elements*; each  $e \in E$  has integral *capacity*  $u_e > 0$ . We are also given a family  $\mathcal{P}$  of subsets of  $E$ , where each  $P \in \mathcal{P}$  is called a *path* and has a linear order  $<_P$  on its elements and *weight* (or *reward*)  $r_P \geq 0$ . (If  $P$  and  $Q$  are two paths both containing  $e$  and  $f$ , it is entirely possible that  $e <_P f$  but  $e >_Q f$ .) If  $e, f \in P$  with  $e <_P f$  then we define, e.g.,  $(e, f]_P = \{g \in P \mid e <_P g \leq_P f\}$ . It is convenient to define artificial elements  $s, t \notin E$  with  $u_s = u_t = \infty$ , such that  $s$  (resp.  $t$ ) is the first (resp. last) element on every  $P \in \mathcal{P}$ . Then we further define, e.g.,  $(s, e)_P = \{g \in P \mid g <_P e\}$ , and  $[e, t)_P = \{g \in P \mid e \leq_P g\}$ .

If  $e \in P \cap Q$  (paths  $P$  and  $Q$  *cross* at  $e$ ), define  $(P, e, Q) = (s, e]_P \cup [e, t)_Q$ . Further define  $P \times_e Q$  to be some member of  $\operatorname{argmax}\{r_R \mid R \subseteq (P, e, Q), R \in \mathcal{P}\}$ . We then require for all  $P, Q \in \mathcal{P}$  with  $e \in P \cap Q$  that

$$r_{P \times_e Q} + r_{Q \times_e P} \geq r_P + r_Q, \tag{1}$$

a sort of supermodularity. Note that the case where  $r_P = 1$  for all  $P \neq \emptyset$  satisfies (1) precisely when  $\mathcal{P}$  satisfies: There is a path contained in  $(P, e, Q)$  (and so also in  $(Q, e, P)$ ). See [6] for other interesting examples of this model.

The *Weighted Abstract Flow (WAF)* problem associated with  $E$  and  $\mathcal{P}$  puts a flow variable  $x_P$  on each  $P \in \mathcal{P}$  and a weight  $y_e$  on each element  $e \in E$ . The dual linear programs are:



$$\begin{array}{ll}
(\mathbf{P}(r)) & \max \sum_P r_P x_P \\
& \text{s.t. } \sum_{P \ni e} x_P \leq u_e \quad \forall e \in E \\
& x \geq 0 \\
(\mathbf{D}(r)) & \min \sum_e u_e y_e \\
& \text{s.t. } \sum_{e \in P} y_e \geq r_P \quad \forall P \in \mathcal{P} \\
& y \geq 0
\end{array}$$

Here  $(\mathbf{P}(r))$  wants to maximize the weighted sum of flows on all paths, the *value* of  $x$ , denoted  $\text{val}(x)$ , subject to the total flow through any element not being more than its capacity, whereas  $(\mathbf{D}(r))$  wants a minimum capacity weighting of elements that covers all paths. We globally assume that  $u$  and  $r$  are integral.

**Theorem 1 (Hoffman, [6, Theorem 2.4]).** *When  $u$  and  $r$  are integral vectors, linear programs  $(\mathbf{P}(r))$  and  $(\mathbf{D}(r))$  have integral optimal solutions.*  $\blacksquare$

Section 2 sets notation for the formal model and the oracle used to access it, and then Section 3 develops the successive shortest paths (SSP) framework of the algorithm. This motivates considering an *Abstract Max Flow/Min Cut (AMFMC)* subproblem restricted to have some elements tight to their capacities, which is solved in Section 4. Then Section 5 embeds this subroutine into the SSP framework to get a pseudopolynomial WAF algorithm, and Section 6 shows how to use this algorithm plus capacity scaling plus sensitivity analysis to get a polynomial WAF algorithm. This extended abstract omits most proofs, figures, and formal algorithms for space reasons.

## 2 Preliminaries

Let  $\mathbb{1}$  denote the vector of all ones. Any  $x \in \mathbb{R}^{\mathcal{P}}$  is a *path-flow*. Every path-flow induces a flow through each element, which we denote as  $x(e) := \sum_{P \ni e} x_P$ . Thus  $x$  is feasible iff  $x \geq 0$  and  $x(e) \leq u_e$  for all  $e \in E$ . If  $x(e) = u_e$  then we say that  $e$  is *saturated* (w.r.t.  $x$ ), and we put  $S(x) := \{e \in E \mid e \text{ is saturated w.r.t. } x\}$  (this is nearly always w.r.t. the current  $x$ , and then we write just “ $S$ ”). If  $x_P > 0$  then we say that  $P$  is *positive*. Any  $y \in \mathbb{R}^E$  is a *dual vector*. Each such  $y$  induces a dual weight on each path, which we denote as  $y(P) := \sum_{e \in P} y_e$ . Thus  $y$  is dual feasible iff  $y \geq 0$  and  $y(P) \geq r_P$  for all  $P \in \mathcal{P}$ . We imagine all paths as running from  $s$  at the extreme left to  $t$  at the extreme right. Hence the “left-most” element of  $P$  with some property means the element of  $P$  with the property that is minimum w.r.t.  $<_P$ , and similarly for “right-most”.

Suppose that  $x$  is a feasible path-flow. Then  $x$  and  $y \geq 0$  are jointly optimal to  $(\mathbf{P}(r))$  and  $(\mathbf{D}(r))$  iff the following conditions hold: (OPT i)  $y(P) \geq r_P$  for all  $P \in \mathcal{P}$ ; (OPT ii)  $x_P \cdot (y(P) - r_P) = 0$  for all  $P \in \mathcal{P}$ ; and (OPT iii)  $y_e \cdot (u_e - x(e)) = 0$  for all  $e \in E$ . Condition (OPT i) is just dual feasibility, and (OPT ii–iii) are complementary slackness.

### 2.1 Accessing the Abstract Network Via an Oracle

We take the point of view that  $|E|$  is “small”, and so we explicitly store  $y$  as a vector in  $\mathbb{R}^E$ . Denote  $m := |E|$ . Let  $n$  denote an upper bound on path length, so that  $\max_{P \in \mathcal{P}} |P| \leq n$ . We think of  $|\mathcal{P}|$  as being “big”, possibly exponential in  $m$ , so we do not have an explicit list of paths and their  $r$ -values. We represent WAF with an oracle

for  $(D(r))$  that when given a vector  $y$ , either verifies that  $y$  is feasible, or returns a constraint violated by  $y$ . In Ellipsoid terms, it is a separation routine (see [5]). We use this oracle to generate interesting paths in  $\mathcal{P}$  as needed. We keep only a list of the current positive paths and their respective  $x$  values. We show bounds on the number of positive paths generated by our algorithm. A “small”  $E$  and “big”  $\mathcal{P}$  is consistent with most applications of this model. Formally the oracle is:

$\mathcal{O}(y)$  when given  $y \in \mathbb{R}_+^E$ , returns either a *violating path*  $P$  with  $y(P) < r_P$  (together with  $r_P$  and  $<_P$ ), or the statement that every  $P \in \mathcal{P}$  satisfies dual feasibility, i.e., (OPT i).

Let PO denote the time for one call to  $\mathcal{O}(y)$ . The results in [5] already imply that  $\mathcal{O}(y)$  is strong enough to get a strongly polynomial algorithm for WAF. However, Ellipsoid cannot produce *integral* optimal solutions to the dual in TDI problems, i.e., integral optimal flows  $x$  for  $(P(r))$ .

Let  $U$  be an upper bound on the  $u_e$  and set  $r_{\max} = \max_{P \in \mathcal{P}} r_P$ . Then the size of the data is  $\log U + \log r_{\max}$ . A pseudopolynomial bound can involve  $U, r_{\max}, m, n$ , and PO, a weakly polynomial bound can involve  $\log U, \log r_{\max}, m, n$ , and PO, and a strongly polynomial bound can involve only  $m, n$ , and PO.

### 3 The Successive Shortest Path Framework

We adapt the Successive Shortest Path (SSP) algorithm for WAF. We start by pushing flow on paths with value  $r_{\max}$ . Let  $\lambda$  be a scalar parameter whose value represents the reward of the current set of paths we are considering. Hence we initialize  $\lambda = r_{\max}$ , and try to drive  $\lambda$  towards zero. A key idea is to relax dual feasibility by  $\lambda$  so that (OPT i) becomes

$$y(P) \geq r_P - \lambda \quad \text{for all } P \in \mathcal{P}. \tag{2}$$

Define  $\text{gap}_\lambda(P) = y(P) - r_P + \lambda$ ; we often abbreviate this to just  $\text{gap}(P)$ . Then the *relaxed optimality conditions* are: (OPT( $\lambda$ ) i)  $\text{gap}(P) \geq 0$  for all  $P \in \mathcal{P}$ ; (OPT( $\lambda$ ) ii)  $x_P \cdot \text{gap}(P) = 0$  for all  $P \in \mathcal{P}$ ; and (OPT( $\lambda$ ) iii)  $y_e \cdot (u_e - x(e)) = 0$  for all  $e \in E$ . Note that  $\lambda = r_{\max}, y \equiv 0$  and  $x \equiv 0$  satisfy (OPT( $\lambda$ ) i–iii). At a general step of the algorithm we have  $x, y$ , and  $\lambda$  such that  $x$  is primal feasible, and  $x$  and  $y$  satisfy (OPT( $\lambda$ ) i–iii). Define the (abstract)  $\lambda$ -*subnetwork* as the abstract network containing paths  $\mathcal{P}(\lambda) := \{P \in \mathcal{P} \mid \text{gap}(P) = 0\}$ . We need to know that the  $\lambda$ -subnetwork is a valid abstract network, i.e., that it satisfies  $\square$ .

**Lemma 2.** *Path system  $\mathcal{P}(\lambda)$  satisfies  $\square$ .*

**Corollary 3.** *For  $P, Q \in \mathcal{P}(\lambda)$  with  $e \in P \cap Q$ ,*

$$r_{P \times_e Q} + r_{Q \times_e P} = r_P + r_Q \quad \text{and} \tag{3}$$

$$y(P \times_e Q) + y(Q \times_e P) = y(P) + y(Q). \tag{4}$$

**Corollary 4.** *Suppose that  $P, Q \in \mathcal{P}(\lambda)$  with  $e \in P \cap Q$ , and  $e <_Q f$ , with  $y_e > 0$ . Then  $e <_{P \times_e Q} f$ .*

We want to augment  $x$  by a max flow in  $\mathcal{P}(\lambda)$ . But (OPT( $\lambda$ )) iii) requires that when  $y_e > 0$ , then  $x(e) = u_e$ . Hence we must find a max flow that preserves that  $x(e) = u_e$  for  $e \in R(y) := \{e \in E \mid y_e > 0\}$ , the *restricted* elements; we usually suppress the dependence on  $y$  and just write “ $R$ ”. The algorithm thus keeps  $R \subseteq S$ . We call this subproblem *Restricted Abstract Max Flow/Min Cut (RAMFMC)*. RAMFMC differs from AMFMC in being given  $R$  and an initial  $x$  such that  $R \subseteq S(x)$ . We solve RAMFMC on  $\mathcal{P}(\lambda)$  as implicitly represented by the oracle.

## 4 Solving Restricted Abstract Max Flow/Min Cut

The dual linear programs for RAMFMC are:

$$\begin{array}{llll}
\max \text{ val}(x) := \mathbb{1}^T x & & \min \text{ cap}(u) := u^T l & \\
\text{s.t. } x(e) \leq u_e & \forall e \in E & \text{s.t. } l(P) \geq 1 & \forall P \in \mathcal{P}(\lambda) \\
x(e) = u_e & \forall e \in R & & \\
x \geq 0 & & l_e \geq 0 & \forall e \in E - R
\end{array} \quad (5)$$

We show below that there is always an optimal  $l$  that is  $0, \pm 1$ . A *signed subset*  $L$  of  $E$  is an ordered pair  $L = (L^+, L^-)$  such that  $L^+, L^- \subseteq E$  and  $L^+ \cap L^- = \emptyset$ . There is a bijection between  $0, \pm 1$  vectors  $l \in \mathbb{R}^E$  and such  $L$ , where  $L^+ = \{e \in E \mid l_e = +1\}$  and  $L^- = \{e \in E \mid l_e = -1\}$ . Note that  $l(P) = |P \cap L^+| - |P \cap L^-|$ . A *restricted abstract min cut (RAMC)* is a signed subset  $L$  such that  $l(P) \geq 1$  for all  $P \in \mathcal{P}(\lambda)$ , and its capacity is  $\text{cap}(L) = u^T l$ . We extend the AMFMC algorithm of [7] to a RAMFMC algorithm for computing optimal  $x$  and  $L$  with  $L^- \subseteq R$  (satisfying the restriction that  $l \geq 0$  on  $E - R$ ) and  $L^+ \subseteq S(x)$ .

A feasible  $x$  and  $l \in \mathbb{R}^E$  with  $l_e \geq 0$ , for all  $e \in E - R$ , are optimal for RAMFMC iff the following conditions hold: (OPT RAMF i)  $l(P) \geq 1$  for all  $P \in \mathcal{P}(\lambda)$ ; (OPT RAMF ii)  $x_P \cdot (l(P) - 1) = 0$  for all  $P \in \mathcal{P}(\lambda)$ ; and (OPT RAMF iii)  $l_e \cdot (u_e - x(e)) = 0$  for all  $e \in E$ . The algorithm tries to construct an  $L$  obeying (OPT RAMF i–iii) and thus proving that  $x$  is optimal. We might find out that  $x$  is not yet optimal by finding a carefully designed subset of paths in  $\mathcal{P}(\lambda)$  by which we can augment  $x$ .

Since RAMFMC is a subproblem of WAF, it is natural to want a representation that uses the same  $\mathcal{O}(y)$  as WAF to check (5). We need an oracle that when given  $l \in \{0, \pm 1\}^E$ , returns either a violating path  $P \in \mathcal{P}(\lambda)$  with  $l(P) < 1$ , or the statement that  $l(P) \geq 1$  for every  $P \in \mathcal{P}(\lambda)$ . The next lemma shows that we can check (5) by calling  $\mathcal{O}((y + \frac{1}{n+1}l)(\lambda - \frac{1}{n+1}))$  (which we call  $\mathcal{O}$ -RAMF( $l$ )), which stands for extending  $y + \frac{1}{n+1}l$  by  $(y + \frac{1}{n+1}l)_s = \lambda - \frac{1}{n+1}$ .

**Lemma 5.** *For  $l \in \{0, \pm 1\}^E$ , and integral  $y, \lambda$  satisfying (OPT( $\lambda$ )) i–iii), it holds that*

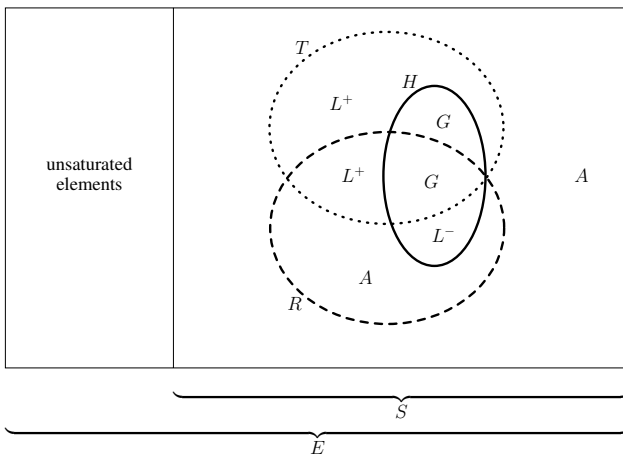
1. *If  $y(P) + \frac{1}{n+1}l(P) \geq r_P - (\lambda - \frac{1}{n+1})$  for all  $P \in \mathcal{P}$ , then  $l$  is a feasible RAMC in  $\mathcal{P}(\lambda)$ .*
2. *If  $y(P) + \frac{1}{n+1}l(P) < r_P - (\lambda - \frac{1}{n+1})$  for some  $P \in \mathcal{P}$ , then  $P$  is in  $\mathcal{P}(\lambda)$  and violates (5), i.e.,  $l(P) \leq 0$  and  $y(P) = r_P - \lambda$ .*

By analogy with ordinary max flow, if we approach  $e$  via a path  $P$  with  $e \in P$  along elements to the left of  $e$  on  $P$  we say that we first reach the “tail” of  $e$ , denoted  $\text{tail}(e)$ ; and if we approach  $e$  via a path  $P$  with  $e \in P$  along elements to the right of  $e$  on  $P$  we say that we first reach the “head” of  $e$ , denoted  $\text{head}(e)$ .

We generalize reachability from  $s$  via a partial augmenting path in ordinary max flow, to reachability from  $s$  via a *partial augmenting structure (PAS)* in RAMFMC. In ordinary max flow, a partial augmenting path traverses some arcs forwards, and some arcs backwards. By aggregating consecutive forwards and backwards arcs into segments, we can consider such a path to consist of alternating positive and negative segments. Each positive segment contains only unsaturated arcs, and each negative segment contains only positive-flow arcs but no restricted arcs.

Thus a first guess at a proper definition of a PAS for abstract flow is that it should be an ensemble of paths  $P_1^+, P_1^-, P_2^+, P_2^-, \dots, P_{k-1}^-, P_k^+$  together with a sequence of elements  $e_1^+, e_1^-, e_2^+, e_2^-, \dots, e_{k-1}^-, e_k^+$  such that (a) each  $e_i^+ \in P_i^+ \cap P_i^-$ ; (b) each  $e_i^- \in P_i^- \cap P_{i+1}^+$ ; (c)  $(e_{i-1}^-, e_i^+)_{P_i^+} \cap S = \emptyset$ ; and (d)  $x_{P_i^-} > 0$  and  $(e_i^+, e_i^-)_{P_i^-} \cap R = \emptyset$ . The intention is that we could feasibly increase flow on each  $(e_{i-1}^-, e_i^+)_{P_i^+}$  (“positive segment”), and decrease flow on each  $(e_i^+, e_i^-)_{P_i^-}$  (“negative segment”).

However, because a cross of two paths may have a different order, we have to modify condition (c) to: for each  $P_i^+, i = 1, \dots, k, e_i^+ \in P_{i-1}^- \times_{e_{i-1}^-} P_i^+$ , and  $e_i^+$  is the left-most (w.r.t.  $<_{P_{i-1}^- \times_{e_{i-1}^-} P_i^+}$ )  $S$ -element in  $(P_{i-1}^- \times_{e_{i-1}^-} P_i^+) \cap (e_{i-1}^-, t)_{P_i^+}$  (i.e., for any  $P_i^-$  containing  $e_i^+$ , we can feasibly increase flow on  $(P_{i-1}^- \times_{e_{i-1}^-} P_i^+) \times_{e_i^+} P_i^-$  if we are also decreasing flow on  $P_{i-1}^-$  and  $P_i^-$ ). When  $e_k^+ = t$ , a PAS to  $\text{tail}(t)$  is called an *augmenting structure (AS)*. Despite our careful definition here, the paths of an AS still



**Fig. 1.** Venn diagram showing the relation between  $R = \{e \mid y_e > 0\}$  (dashed line),  $T$  (dotted line) and  $H$  (solid line) defined by the existence of forward and backward PAS’s, and  $L^+, L^-$ , and  $G$  used to define the current guess at a RAMC

might intersect each other, preventing an integral feasible flow update. Later AUGMENT in Section 4.1 shortcuts such intersections so as to produce feasible augmentations.

Define  $T = \{e \in S \mid \text{there exists a PAS to } \text{tail}(e)\}$  and  $H = \{e \in S \mid \text{there exists a PAS to } \text{head}(e)\}$ . Arc  $i \rightarrow j \in \partial^+(C^*)$  for a left-most min cut  $C^*$  in ordinary max flow if  $s$  can reach  $i$  but not  $j$ , which corresponds to  $L^+ = T - H$ . Arc  $i \rightarrow j \in \partial^-(C^*) \cap R$  for ordinary max flow if  $s$  can reach  $j$  but not  $i$ , which corresponds to  $L^- = (H - T) \cap R$ . If  $e \in H - R$ , then the PAS to  $\text{head}(e)$  can be extended “through”  $e$  to  $\text{tail}(e)$  (including a null  $P^+$  segment from  $\text{tail}(e)$  to  $\text{tail}(e)$ ) to get a PAS to  $\text{tail}(e)$ . Therefore  $H - R \subseteq T$ , so that in fact  $L^-$  is just  $H - T$ . Elements that are in  $H \cap T$  cannot be in  $L^+$  or in  $L^-$ ; we call them *garbage elements* and set  $G = H \cap T$ . Define  $A = S - (G \cup L^+ \cup L^-)$  as the *active elements*. It always holds that  $A \cup L^+ \cup L^- \cup G = S$  is a disjoint partition of  $S$  (see Figure 1).

### 4.1 The RAMFMC Algorithm

The algorithm tries to guess  $L$  satisfying (OPT RAMF ii–iii), and uses  $\mathcal{O}$ -RAMF( $l$ ) to find paths violating (OPT RAMF i).

We use  $P^+(e)$  to denote the last path in a PAS to  $\text{tail}(e)$ , and  $P^-(e)$  to denote the last path in a PAS to  $\text{head}(e)$ . We start with  $A = S$ . When we first discover a new PAS to  $\text{tail}(e)$  we move  $e$  from  $A$  to  $L^+$ . If we later discover a PAS to  $\text{head}(e)$  then we move  $e$  from  $L^+$  to  $G$ . Similarly, when we first discover a new PAS to  $\text{head}(e) \in R$  we move  $e$  from  $A$  to  $L^-$ . If we later discover a PAS to  $\text{tail}(e)$  then we move  $e$  from  $L^-$  to  $G$ . Between AUGMENTS,  $R$ -elements move only from  $A$  to  $L^-$  to  $G$  or from  $A$  to  $L^+$  to  $G$ ; elements in  $S - R$  move only from  $A$  to  $L^+$  to  $G$ , or directly from  $A$  to  $G$ .

In this manner we try to find an AS by gradually growing PAS’s, which increases the number of elements known to be in  $T \cup H$ . One of two things will stop us: (1) A PAS reaches  $t$ , in which case we can augment flow; or (2)  $\mathcal{O}$ -RAMF( $l$ ) reports no paths violating (OPT RAMF i), and so the current  $L$  satisfies (OPT RAMF i–iii), and so it is a RAMC proving that the current  $x$  is optimal. Subroutine IMPROVE implements this idea of growing PAS’s. It has two subroutines itself—REDUCE and AUGMENT.

**The IMPROVE Subroutine.** IMPROVE takes a given feasible  $x$  and tries either to find an AS so it can call AUGMENT to find a new flow of higher value, or to find an  $L$  proving that  $x$  is optimal. There are two processes that affect  $L$ : IMPROVE itself calls  $\mathcal{O}$ -RAMF( $l$ ) to find a path  $P$  violating (OPT RAMF i), i.e., with  $l(P) \leq 0$ , and then it tries to find elements of  $A$  on  $P$  that it can move into  $L^+$  to ensure that  $l(P) \geq 1$ . IMPROVE fails at this only by finding that it can call AUGMENT, which performs shortcuts on the AS so as to integrally augment  $x$ .

However, since  $S$ -elements appear on multiple paths, it is easy for IMPROVE to create a positive path  $P$  with  $l(P) > 1$ , violating (OPT RAMF ii). To deal with this, every time that IMPROVE changes  $L$  it calls REDUCE, which processes every positive path  $P$  to ensure that  $l(P) \leq 1$ ; REDUCE processes even those positive  $P$  with  $l(P)$  already equal to 1 in order to make the properties in Lemma 6 below true. Roughly speaking, REDUCE does this by moving surplus  $L^+$  elements to  $G$ . At the end of the section we embed IMPROVE into an overall algorithm for solving RAMFMC.

If  $\mathcal{O}$ -RAMF( $l$ ) returns violating path  $P$ , then  $P$  must contain some  $e \in (\{s\} \cup H)$ . Now  $e \in (\{s\} \cup H)$  implies that there is at least one PAS to  $\text{head}(e)$ ; let  $Q_e$  be the final positive path in this PAS (if  $e = s$ , then setting  $Q_e = \emptyset$  works together with  $Q_e \times_e P = P$ ). If there is no  $S$ -element in  $(Q_e \times_e P) \cap (e, t)_P$ , then the PAS using  $Q_e$  to  $\text{head}(e)$  together with  $P$  forms an AS and IMPROVE calls AUGMENT.

Otherwise, let  $f$  be the left-most (w.r.t.  $<_{Q_e \times_e P}$ )  $S$ -element on  $(Q_e \times_e P) \cap (e, t)_P$ . Then with  $f = e_i^+$ ,  $e = e_{i-1}^-$ ,  $Q_e = P_{i-1}^-$ ,  $P = P_i^+$ , and any positive path containing  $f$  as  $P_i^-$ , clearly this satisfies part 4 of the PAS definition, and so using  $P$  to extend the PAS to  $\text{head}(e)$  up to  $\text{tail}(f)$  works. Therefore putting  $f$  into  $T$  is correct. We set  $P^+(f) = P$ ,  $\text{pred}(f) = e$ , and  $\text{predpath}(f) = Q_e$  to record that the PAS reached  $\text{tail}(f)$  from the immediate predecessor element  $e$  and path  $Q_e$  for later use in AUGMENT.

**The REDUCE Subroutine.** Suppose that  $P$  is a positive path with  $f \in P \cap L^+$ , and that  $e$  is the right-most  $S$ -element on  $(s, f)_P$ . Since  $f \in L^+$  there is a PAS to  $\text{tail}(f)$ , and since  $P$  is positive we can extend this along  $P$  to  $\text{head}(e)$ , thereby showing that  $e \in H$ . If  $e \notin R$ , then we can extend this PAS to  $\text{tail}(e)$ , showing that  $e \in G$ ; if  $e \in T$ , then (even if  $e \in R$ ) the new PAS to  $\text{head}(e)$  shows again that  $e \in G$ . In either case we have that  $e \in T$ , and so we can reset  $f = e$  and iterate. We set  $P^-(e) = P$  and  $\text{pred}(e) = f$  to record the PAS that put  $e$  into  $T$  for later use in AUGMENT.

This process stops only when it reaches an  $e \in R$  with  $e \notin T$ . In this case,  $e \in R \cap H$  means that we can put  $e$  into  $L^-$ , and we again set  $P^-(e) = P$  and  $\text{pred}(e) = f$ . We then re-start the process with the next  $L^+$  to the left of  $e$  on  $P$ .

Suppose that  $P, Q \in \mathcal{P}(\lambda)$  with  $e \in P \cap Q$ . Because  $P \times_e Q \subseteq (P, e, Q)$ , and  $P \times_e Q$  contains all the  $L^-$  elements of  $(P, e, Q)$  (by Corollary 4), we get

$$l(P \times_e Q) \leq l((s, e)_P) + l_e + l((e, t)_Q). \quad (6)$$

**Lemma 6.** *When IMPROVE returns  $L$ , it is an optimal RAMC. Furthermore, every positive path  $P$  has the following properties:*

1. *The left-most  $L$ -element on  $P$  is in  $L^+$ .*
2. *The right-most  $L$ -element on  $P$  is in  $L^+$ .*
3. *The  $L^+$  and  $L^-$  elements on  $P$  alternate.*
4. *If  $e$  and  $f$  are consecutive  $L$ -elements on  $P$ , then*
  - (a) *if  $f \in L^+$  then  $e \in L^-$  and all  $g \in (e, f)_P \cap S$  are in  $G$ ;*
  - (b) *if  $f \in L^-$  then  $e \in L^+$  and all  $g \in (e, f)_P \cap S$  are in  $A$ .*

**Corollary 7.** *When IMPROVE returns  $L$ , if  $P$  is a positive path containing  $e$ , then*

$$\text{if } e \in L^+ \text{ then } l((s, e]_P) = 1 \text{ and } l((e, t)_P) = 0; \quad (7)$$

$$\text{if } e \in L^- \text{ then } l((s, e]_P) = 0 \text{ and } l((e, t)_P) = 1; \quad (8)$$

$$\text{if } e \in G \text{ then } l((s, e]_P) = 0 \text{ and } l((e, t)_P) = 1; \quad (9)$$

$$\text{if } e \in A \text{ then } l((s, e]_P) = 1 \text{ and } l((e, t)_P) = 0. \quad (10)$$

**The AUGMENT Subroutine.** Since IMPROVE and REDUCE record  $P^+(g)$ ,  $P^-(g)$ ,  $\text{pred}(g)$ , and  $\text{predpath}(g)$  as they construct PAS's, it is fairly easy to follow these pointers backwards to  $s$  to reconstruct the full AS from  $s$  to  $t$ .

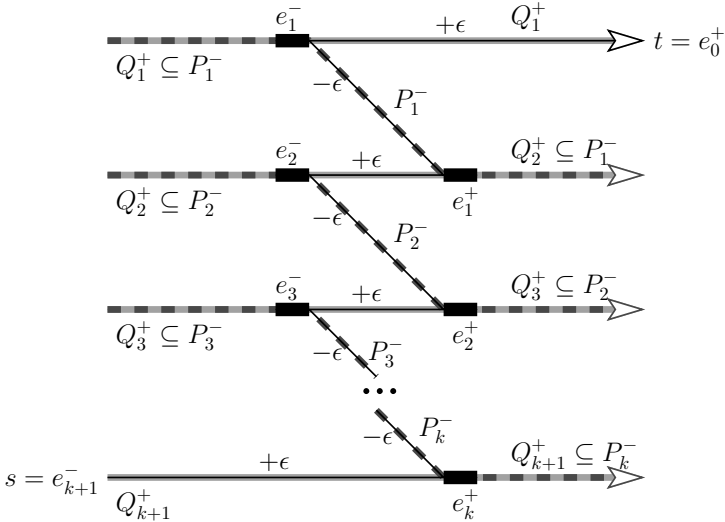
Each  $e_{j-1}^+ \in T$ , and so  $P_j^+ = P^+(e_{j-1}^+)$  exists, and it contains the predecessor  $e_j^- = \text{pred}(e_{j-1}^+) \in H$ . Due to how IMPROVE constructed a PAS to  $\text{tail}(e_{j-1}^+)$ , there is also a predecessor path  $P_j^- = \text{predpath}(e_{j-1}^+)$ . This is constructed such that the path  $Q_j^+ := (P_j^- \times_{e_j^-} P_j^+) \times_{e_{j-1}^+} P_{j-1}^-$  is well-defined. Notice that when  $j = 1$ , then  $e_0^+ = t$  and we set  $P_0^- = \emptyset$ , and interpret  $Q_1^+ = (P_1^- \times_{e_1^-} P_1^+) \times_t P_0^-$  as just  $P_1^- \times_{e_1^-} P_1^+$ . This process of chasing pointers backwards ends once some  $e_{k+1}^- = s$ . In this case we could think of  $P_{k+1}^- = \emptyset$ , so that  $Q_{k+1}^+ = (P_{k+1}^- \times_s P_{k+1}^+) \times_{e_k^+} P_k^- = P_{k+1}^+ \times_{e_k^+} P_k^-$ .

Naively we then want to increase flow by  $\epsilon$  on each  $Q_j^+$  and decrease flow by  $\epsilon$  on each  $P_j^-$ . This has the effect of increasing  $x(e)$  for  $e$  in  $Q_j^+$ 's positive segment ( $Q_j^+ - P_j^- - P_{j-1}^-$ ), and decreasing  $x(e)$  for  $e$  in  $P_j^-$ 's negative segment  $P_j^- - Q_j^+ - Q_{j+1}^+$ . For the correctness theorem we need some lemmas.

**Lemma 8.** For any two paths  $P, Q \in \mathcal{P}(\lambda)$  with  $e' \in P \cap Q$  each  $e \in R$  must appear with the same multiplicity in  $P \times_{e'} Q$  and  $Q \times_{e'} P$  as it appears with in  $P$  and  $Q$ .

**Lemma 9.** For each  $j$ ,  $Q_j^+$ 's positive segment contains no  $S$ -elements, and  $P_j^-$ 's negative segment contains no  $R$ -elements.

Thus if we decrease flow on  $P_j^-$  and  $P_{j-1}^-$  by  $\epsilon = 1$ , and increase flow on  $Q_j^+$  by  $\epsilon$ , then this cannot cause the flow through any  $e \in S$  to become larger than  $u_e$ . Figure 2 schematically indicates this by the notation that  $Q_j^+ \subseteq P_j^-$  on  $(s, e_j^-)_{P_j^-}$ , and  $Q_j^+ \subseteq$



**Fig. 2.** A schematic picture of the paths built in AUGMENT before doing shortcuts. All gray paths are  $Q_j^+$ 's, all dashed paths are  $P_j^-$ 's. The thin black line indicates where the actual flow change happens. Segments are marked with their net flow change of  $+\epsilon$  or  $-\epsilon$ . All  $e_j^-$ 's are in  $H$ , whereas the  $e_j^+$ 's are in  $T$ . Note that this picture is simplified as paths may overlap arbitrarily and the order of elements on paths sharing a common segment might differ.



$P_{j-1}^-$  on  $(e_{j-1}^-, t)_{P_{j-1}^-}$ ; however, in fact the order of elements on  $Q_j^+$  might be different from the orders on  $P_j^-$  and  $P_{j-1}^-$ , and so this should not be taken too literally. Similarly, the local flow change that decreases flow on  $P_j^-$  by  $\epsilon = 1$  and increases flow on  $Q_j^+$  and  $Q_{j+1}^+$  by  $\epsilon$  does not de-saturate any  $e \in R$ .

**Lemma 10.** *For  $j = 1, \dots, k$ , all  $e_j^-$  found in AUGMENT are pairwise distinct, and all  $e_j^+$  are pairwise distinct (however, we could have  $e_j^+ = e_i^-$  for some  $i$  and  $j$ ).*

**The SHORTCUT Subroutine.** Although Lemma 9 shows that our proposed flow change is locally feasible, it might not be globally feasible. It could happen that the positive segments from  $Q_j^+$  and  $Q_h^+$  intersect, or that  $P_j^- = P_h^-$  for some  $j \neq h$ . AUGMENT calls SHORTCUT to deal with these possibilities. It uses the oracle to cross paths and hence shortcut the PAS. Then it computes a positive, integral flow increment  $\epsilon$ , and augments flows by  $\epsilon$ . Since there is exactly one more positive segment than negative segments,  $\mathbb{1}^T x$  increases by  $\epsilon$ .

**Lemma 11.** *AUGMENT runs in  $O(m(mn + \text{PO}))$  time.*

**Theorem 12.** *For a given feasible integral restricted flow  $x$  (and integral capacities), the flow returned by AUGMENT is a feasible integral restricted flow of strictly larger value.*

**Putting the Pieces Together.** We can now embed IMPROVE in the actual RAMFMC algorithm that computes a restricted abstract max flow and a corresponding RAMC. We then go through a series of lemmas establishing the correctness and running times of the various pieces.

**Corollary 13.** *REDUCE runs in  $O(m^2 nU)$  time.*

**Corollary 14.** *IMPROVE runs in  $O(m^3 nU + m\text{PO})$  time.*

**Theorem 15.** *The RAMFMC Algorithm returns a restricted abstract max flow  $x$  and a restricted abstract min cut  $L$  such that  $\text{val}(x) = \text{cap}(L)$  in  $O(mU(m^3 nU + m\text{PO}))$  time.*

## 5 The WAF Algorithm

Recall that our general idea for the WAF algorithm is to relax dual feasibility to 2 and to gradually drive  $\lambda$  down from  $r_{\max}$  to 0. Given the current  $\lambda$ -subnetwork  $\mathcal{P}(\lambda)$  along with a primal feasible flow  $x$ , the WAF Algorithm calls the RAMFMC Algorithm to compute optimal  $x'$  and  $L$  (and its corresponding  $l$ ). It replaces  $x$  by  $x'$ , and then updates to  $y' = y + \theta l$  and  $\lambda' = \lambda - \theta$  for some step length  $\theta$ .

### 5.1 Computing Step Length $\theta$

Three factors constrain the choice of  $\theta$ :

- [1] The need to keep  $y' \geq 0$ . This means that  $\theta \leq \min\{y_e \mid e \in L^-\}$ .
- [2] The need to keep  $\lambda' \geq 0$ . This means that  $\theta \leq \lambda$ .



[3] The need to preserve (OPT( $\lambda$ ) i) as  $y$  and  $\lambda$  change. This means that  $(y + \theta l)(P) \geq r_P - (\lambda - \theta)$  for each  $P \in \mathcal{P}$ , or  $\theta \leq \text{gap}(P)/(1 - l(P))$  for all  $P$  with  $l(P) \leq 0$  (note that  $\text{gap}(P)$  is computed w.r.t.  $y$  and  $\lambda$ ; we use  $\text{gap}'(P)$  when computed w.r.t.  $y'$  and  $\lambda'$ ).

Every  $P$  with  $x'_P > 0$  is in  $\mathcal{P}(\lambda')$ . Lemma 6 ensures that  $x'_P > 0$  implies that  $l(P) = 1$ , and therefore  $y'(P) - y(P) = \theta = \lambda - \lambda'$ , or  $\text{gap}'(P) = \text{gap}(P) = 0$ , and so (OPT( $\lambda$ ) ii) is maintained. Moreover, (OPT( $\lambda$ ) iii) is preserved, because every element  $e$  with  $y_e > 0$  is kept saturated.

Define  $\theta = \min(\lambda, \min\{y_e \mid e \in L^-\})$  as the upper bound on  $\theta$  coming from [1] and [2], which is trivial to compute in  $O(m)$  time. The bound from [3] is trickier since  $|\mathcal{P}|$  is “large”, and so needs to rely on  $\mathcal{O}(y)$ . The next two lemmas show that in [3] it suffices to consider only paths  $P$  with  $l(P) = 0$ .

**Lemma 16.** *Suppose that we update  $y$  to  $y'$  via  $\theta$  where*

$$\theta = \min\{\text{gap}(P)/(1 - l(P)) \mid l(P) \leq 0\} < \min\{y_e \mid e \in L^-\}, \quad (11)$$

*i.e., that  $\theta$  is determined by [3] and not by [1]. Further suppose that  $\hat{P}, Q \in \mathcal{P}(\lambda')$  with  $x_Q > 0$ . Then*

$$l(\hat{P}) + l(Q) = l(\hat{P} \times_e Q) + l(Q \times_e \hat{P}). \quad (12)$$

**Lemma 17.** *If (11) is true, then this minimum is attained at some  $P$  with  $l(P) = 0$ .*

Thus [3] becomes  $\theta \leq \text{gap}(P)$  for  $P$  such that  $l(P) = 0$ , proving that  $\theta$  is an integer, and so  $y'$  and  $\lambda'$  are integral. To compute  $\theta$ , we take advantage of knowing that it is an integer. First call  $\mathcal{O}(y + \bar{\theta}l \mid \lambda - \bar{\theta})$ . If there are no violating paths, then we can just set  $\theta = \bar{\theta}$ . Otherwise, the new value of  $\theta$  is the largest  $\hat{\theta} \in (0, \bar{\theta})$  such that  $\mathcal{O}(y + \hat{\theta}l \mid \lambda - \hat{\theta})$  returns no violating path. This can be determined via binary search with  $O(\log r_{\max})$  calls to the oracle. Since computing  $y + \hat{\theta}l$  costs  $O(m)$  for each  $\hat{\theta}$ , the whole process of determining the step length takes  $O(\log r_{\max}(m + \text{PO}))$  time.

**Lemma 18.** *These primal and dual updates preserve (OPT( $\lambda$ ) i–iii).*

## 5.2 WAF Algorithm Running Time

It is easy to construct examples where  $x$  is not augmented during an iteration (but  $y$  and  $\lambda$  are changed). Suppose that an iteration changes  $y, \lambda$  to  $y', \lambda'$ , but does not change  $x$ . Let  $R$  be defined w.r.t.  $y$  and  $R'$  w.r.t.  $y'$ , and let  $T$  and  $H$  denote elements whose heads and tails are reachable via PAS's in  $\mathcal{P}(\lambda)$  using  $x$  and  $R$ , and  $T'$  and  $H'$  denote the same in  $\mathcal{P}(\lambda')$  using  $x$  and  $R'$ .

**Lemma 19.** *If a non-terminal iteration changes  $y, \lambda$  to  $y', \lambda'$  but does not change  $x$ , then*

$$T \subseteq T' \text{ and } H \subseteq H'. \quad (13)$$

**Lemma 20.** *Suppose that a non-terminal iteration changes  $y$  to  $y'$  but does not change  $x$ . Then at least one inclusion in (13) is proper.*

**Theorem 21.** *The WAF algorithm runs in  $O(\min\{r_{\max}, m^2U\} \cdot (\log r_{\max}(m + \text{PO}) + mU(m^3nU + m\text{PO})))$  time, and the final  $x$  and  $y$  are integral optimal solutions.*

Notice that Theorem 21 also gives a constructive proof of Theorem 1. The running time bound of Theorem 21 is only pseudopolynomial, as it depends on  $r_{\max}$  and  $U$ . One can adapt a construction of Zadeh [12] to demonstrate that the dependence on  $r_{\max}$  cannot be removed. Zadeh constructed a family of instances of min-cost flow parametrized by  $k$  such that instance  $k$  has  $O(k)$  nodes,  $O(k^2)$  arcs, and SSP takes  $O(2^k)$  iterations.

So far we have dealt with the objective to maximize the total weight of a flow, regardless of its actual flow value. However, it is easy to adapt our algorithm to the case where we want a flow that routes the maximum number of units possible and whose weight is maximum among those flows. We call such a flow a *maximum abstract flow at maximum weight (MAFMW)*.

## 6 A Polynomial Capacity-Scaling WAF Algorithm

The seminal paper by Edmonds and Karp [2] pointed out that scaling is a useful technique for turning algorithms that augment a feasible solution into algorithms that can find an optimal solution in weakly polynomial time. For example, [11] uses scaling to show that augmentation in strongly polynomial time and optimization in strongly polynomial time are equivalent for 0–1 optimization problems. When the number of positive paths is strongly polynomial, our IMPROVE routine is strongly polynomial, but our problem is not 0–1, so we cannot use the results in [11].

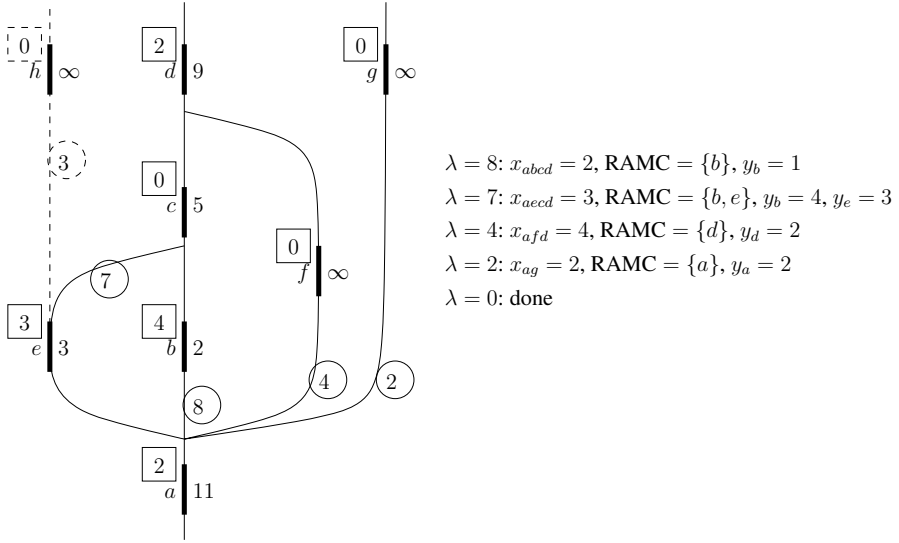
In theory we could choose to scale either the rewards  $r_P$  or the capacities  $u_e$ . It is not clear how to scale the  $r_P$  so that (1) is preserved. Hence we scale the capacities. Recall that  $U$  is the largest capacity of any element, and set  $b = \lfloor \log_2 U \rfloor$  (one less than the number of bits needed to represent the largest capacity). For  $0 \leq k \leq b + 1$ , define  $u_e^k = \lfloor u_e / 2^k \rfloor$ , the  $k$ -th scale of  $u$ . Thus  $u^{b+1} = 0$ ,  $u^b$  is a 0–1 vector, and  $u^0 = u$ .

Since the  $u^b$  instance has  $U = 1$ , Theorem 21 shows that the algorithm computes  $x$  and  $y$  optimal for  $u^b$  in  $O(\min\{r_{\max}, m^2\} \cdot (\log r_{\max}(m + \text{PO}) + m^4n + m^2\text{PO}))$  time. Clearly  $2x$  and  $y$  are optimal for capacities  $2u^b$ . Note that  $2u^b$  is “close” to  $u^{b-1}$  in the sense that  $u_e^{b-1} - 2u_e^b$  is 0 or 1 for all  $e \in E$ . Define the set of elements whose capacities need to be incremented at scale  $k$  as  $I(k) = \{e \mid u_e^{k-1} - 2u_e^k = 1\}$ , and  $\chi(e) \in \mathbb{R}^E$  as the characteristic vector of  $e$ .

We would like to replace  $u' := 2u^b$  by  $u' + \chi(e)$  for each  $e \in I(b)$  and then modify the optimal  $x, y$  for  $u'$  to a new optimal  $x', y'$  for  $u' + \chi(e)$ . The next section develops a subroutine  $\text{REOPT}(e; r, u)$  which does this. Thus if we call  $\text{REOPT}$  for each  $e \in I(b)$  in turn, we will get  $x$  and  $y$  optimal for  $u^{b-1}$ , and we can continue like this until we get  $x$  and  $y$  optimal for  $u^0 = u$ .

### 6.1 Solving the Incremental Capacity Subproblem

The input to  $\text{REOPT}$  is an instance with rewards  $r$  and capacities  $u$ ,  $x$  and  $y$  optimal for  $r$  and  $u$ , and some  $e \in E$ . The goal is to modify  $x$  and  $y$  so they are optimal for  $r$  and  $u + \chi(e)$ . Notice that if  $y_e = 0$ , then  $x' = x$  and  $y' = y$  are again optimal for  $u' + \chi(e)$ ,



**Fig. 3.** An example of the WAF Algorithm in action. Reward values are in circles next to paths, so that, e.g.,  $r_{aecd} = 7$ . For each element  $e$ , the value of  $u_e$  is to the right of  $e$ , and the optimal value of  $y_e$  is in a box above and to the left of  $e$ . The dashed path  $ae h$  does not belong to  $\mathcal{P}(0)$  since  $\text{gap}(ae h) = 2 > 0$ . If we call IMPROVE on  $\mathcal{P}(0)$  it returns  $L = (\{a\}, \emptyset)$ .

and so the algorithm first deletes all such elements from  $I(b)$ . In theory we can use  $y_e$  as a guide: as the dual variable corresponding to the  $x(e) \leq u_e$  constraint, it should tell us how the optimal objective value changes when we replace  $u$  by  $u + \chi(e)$ . In practice, degeneracy often implies that the change in optimal objective value is different from  $y_e$ . REOPT changes the given dual optimal  $y$  into an alternate dual optimal solution as necessary to be able to change  $x$  while maintaining complementary slackness.

We now change flows on two *generalized PAS's* (gPAS's). Let us first consider a simple case. Suppose that there is a PAS  $C^{\text{tail}}$  to  $\text{tail}(e)$  in  $\mathcal{P}(\lambda)$  w.r.t.  $x$  and  $y$ , i.e., that IMPROVE returns RAMC  $L$  with  $e \in T$ . Notice that it is easy to reverse everything in IMPROVE such that it searches for PAS's from  $\text{head}(f)/\text{tail}(f)$  to  $t$ , instead of from  $s$ ; call this REVIMPROVE; such a "reversed" PAS is one type of gPAS. Further suppose that REVIMPROVE finds a PAS  $C^{\text{head}}$  from  $\text{head}(e)$  to  $t$ . Then we could glue together  $C^{\text{tail}}$ ,  $e$ , and  $C^{\text{head}}$  into an AS such that we could call AUGMENT using  $u + \chi(e)$  and it would change  $x$  into  $x'$  such that  $x'(e) = x(e) + 1$ . Then it is easy to see that  $x'$  and  $y$  are jointly optimal w.r.t.  $u + \chi(e)$ .

However, it often happens that  $L$  blocks any PAS from  $s$  from reaching  $\text{tail}(e)$ . For the example in Figure 3,  $L$  prevents a PAS from  $s$  to the tail of every element other than  $a$ . In this example, suppose we are interested in  $u + \chi(d)$ . Then the correct flow change is to push one unit of flow through the gPAS  $C^{\text{tail}}$  that uses  $(a, t)_{ag}$  backwards, then uses  $(a, d)_{afd}$  forwards to  $\text{tail}(d)$ , together with the trivial reverse PAS from  $\text{head}(d)$  to  $t$ ; this is a "cycle" from  $t$  to  $t$  containing  $d$ .

Note that in general, gPAS's to the tail of our key element such as  $C^{\text{tail}}$  that "start from  $t$ " follow the rules for a PAS from  $s$ . To find such gPAS's we adapt IMPROVE to

a *Phase II* version with this trick: set  $t \in L^+$  in the initialization, which is equivalent to adding new path  $st$  to  $\mathcal{P}$ ; we call ordinary IMPROVE without this initialization *Phase I IMPROVE*. In Phase II IMPROVE we also skip the check for an AS (because putting  $t \in L^+$  otherwise creates spurious AS's).

**Lemma 22.** *If  $y_e > 0$ , then either Phase I or Phase II IMPROVE finds a gPAS to  $\text{tail}(e)$ .*

Often neither REVIMPROVE nor Phase II REVIMPROVE (which artificially puts  $s$  into  $L^+$ ) can find a gPAS from  $\text{head}(e)$  to  $t$ . Let  $l^t$  denote  $l$  with the extra  $l_s^t = +1$  component. By (OPT RAMF i)  $l^t(P) \geq 1$  for all  $P \in \mathcal{P}(0)$ , implying that  $l(P) \geq 0$  (since all  $P$  contain  $s$ ). In such cases we treat  $l$  as a *direction vector*, and consider the move  $y' = y + \alpha l$  for some step length  $\alpha \geq 0$ . Two factors determine  $\alpha$ : (a) To keep  $y' \geq 0$  we must have that  $\alpha \leq \min\{y_f \mid f \in L^-\}$ ; since  $f \in L^-$  implies that  $y_f > 0$ , this min is positive. (b) To keep feasible we must have that  $\alpha \leq \min\left\{\frac{-\text{gap}_0(P)}{l(P)} \mid P \text{ s.t. } l(P) < 0\right\}$ ; since  $l(P) \geq 0$  for  $P \in \mathcal{P}(0)$ , all  $P$  with  $l(P) < 0$  have  $\text{gap}_0(P) > 0$ , and so this min is also positive. A lemma similar to Lemma 17 shows that when (b) determines  $\alpha$ , it does so at some  $P$  with  $l(P) = -1$ , and so  $\alpha$  is a positive integer. A binary search similar to the computation of  $\theta$  can be used to compute  $\alpha$  in  $O(\log r_{\max}(m + \text{PO}))$  time. Changing to  $y'$  causes all  $P$  with  $l(P) > 0$  to leave  $\mathcal{P}(0)$ , but this is okay since  $l(P) > 0$  means that  $x_P = 0$ . If  $\alpha$  is determined by (a), changing to  $y'$  would mean that  $y'_e = 0$  for some  $e$ 's. If any such  $e$ 's are in  $I(b)$ , we just remove them from  $I(b)$ . If  $\alpha$  is determined by (b), changing to  $y'$  causes some new  $P$ 's to enter  $\mathcal{P}(0)$ .

**Lemma 23.** *When we set  $y' = y + \alpha l$  as above,  $y'$  is an alternate optimal dual solution.*

Using arguments similar to Lemma 20 we can prove that after  $O(m)$  such dual changes, either  $y_e$  becomes zero (in which case no primal change is necessary), or a gPAS from  $\text{head}(e)$  appears. Then we AUGMENT along the two gPAS's to produce a new optimal  $x$ .

## 6.2 The Capacity-Scaling Algorithm

With REOPT in place, it is now easy to write the overall capacity-scaling algorithm and derive its running time.

**Theorem 24.** *The capacity-scaling algorithm solves WAF in  $O(\log r_{\max}(m^3 + m^2\text{PO}) + m^6n + m^4\text{PO})$  time.*

This is a (weakly) polynomial algorithm for WAF, as compared to the original WAF algorithm, which is only pseudopolynomial.

## References

1. Applegate, D.L., Cook, W.J., McCormick, S.T.: Integral Infeasibility and Testing Total Dual Integrality. OR Letters 10, 37–41 (1991)
2. Edmonds, J., Karp, R.M.: Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. J. ACM 19, 248–264 (1972)

3. Ford Jr., L.R., Fulkerson, D.R.: Maximal Flow through a Network. *Canadian J. of Mathematics* 8, 399–404 (1956)
4. Frank, A., Tardos, É.: An Application of Simultaneous Diophantine Approximation in Combinatorial Optimization. *Combinatorica* 7, 49–65 (1987)
5. Grötschel, M., Lovász, L., Schrijver, A.: *Geometric Algorithms and Combinatorial Optimization*. Springer, Heidelberg (1988)
6. Hoffman, A.J.: A Generalization of Max Flow-Min Cut. *Math. Prog.* 6, 352–359 (1974)
7. McCormick, S.T.: A Polynomial Algorithm for Abstract Maximum Flow. UBC Faculty of Commerce Working Paper 95-MSC-001. In: An extended abstract appeared in *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 490–497 (1995)
8. McCormick, S.T.: Submodular Function Minimization. In: Aardal, K., Nemhauser, G., Weismantel, R. (eds.) *Handbook on Discrete Optimization* Ch. 7, pp. 321–391. Elsevier, Amsterdam (2006)
9. McCormick, S.T., Fujishige, S.: Strongly Polynomial and Fully Combinatorial Algorithms for Bisubmodular Function Minimization. In: *Mathematical Programming; an extended abstract appeared in Proceedings of Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 44–53 (submitted, 2008)
10. Schrijver, A.: *Theory of Linear and Integer Programming*. John Wiley & Sons, New York (1986)
11. Schulz, A.S., Weismantel, R., Ziegler, G.M.: 0/1-Integer Programming: Optimization and Augmentation are Equivalent. Technical Report No.441/1995, Fachbereich Mathematik, Technische Universität Berlin (1995)
12. Zadeh, N.: A bad network problem for the simplex method and other minimum cost flow algorithms. *Mathematical Programming* 5, 255–266 (1973)

# A Comparative Study of Linear and Semidefinite Branch-and-Cut Methods for Solving the Minimum Graph Bisection Problem

Michael Armbruster<sup>1</sup>, Marzena Fügenschuh<sup>2</sup>, Christoph Helmberg<sup>1</sup>,  
and Alexander Martin<sup>2</sup>

<sup>1</sup> Chemnitz University of Technology

Department of Mathematics, D-09107 Chemnitz, Germany

helmberg@mathematik.tu-chemnitz.de

<sup>2</sup> Technische Universität Darmstadt

Department of Mathematics, D-64289 Darmstadt, Germany

martin@mathematik.tu-darmstadt.de

**Abstract.** Semidefinite relaxations are known to deliver good approximations for combinatorial optimization problems like graph bisection. Using the spectral bundle method it is possible to exploit structural properties of the underlying problem and to apply, even to sparse large scale instances, cutting plane methods, probably the most successful technique in linear programming. We set up a common branch-and-cut framework for linear and semidefinite relaxations of the *minimum graph bisection problem*. It incorporates separation algorithms for valid inequalities presented in the recent study [2] of the facial structure of the associated polytope. Extensive numerical experiments show that the semidefinite branch-and-cut approach outperforms the classical simplex approach on a clear majority of the sparse large scale test instances. On instances from compiler design the simplex approach is faster.

**Keywords:** Branch and cut algorithms, cutting plane algorithms, polyhedral combinatorics, semidefinite programs.

## 1 Introduction

Let  $G = (V, E)$  be an undirected graph with  $V = \{1, \dots, n\}$  and  $E \subseteq \{\{i, j\} : i, j \in V, i < j\}$ . For given vertex weights  $f_v \in \mathbb{N} \cup \{0\}$ ,  $v \in V$ , and edge costs  $w_{\{i, j\}} \in \mathbb{R}$ ,  $\{i, j\} \in E$ , a partition of the vertex set  $V$  into two disjoint clusters  $S$  and  $V \setminus S$  with sizes  $f(S) \leq F$  and  $f(V \setminus S) \leq F$ , where  $F \in \mathbb{N} \cap [\frac{1}{2}f(V), f(V)]$ , is called a *bisection*. Finding a bisection such that the total cost of edges in the cut  $\delta(S) := \{\{i, j\} \in E : i \in S \wedge j \in V \setminus S\}$  is minimal is the *minimum bisection problem* (MB). The problem is known to be NP-hard [9]. The polytope associated with MB,

$$P_B := \text{conv} \left\{ y \in \mathbb{R}^{|E|} : y = \chi^{\delta(S)}, S \subseteq V, f(S) \leq F, f(V \setminus S) \leq F \right\},$$

where  $\chi^{\delta(S)}$  is the incidence vector of the cut  $\delta(S)$  with respect to the edge set  $E$ , is called the MB as well as  $P_B$  are related to other problems and polytopes already known in the literature. Obviously, the bisection cut polytope is contained in the [\[36\]](#)

$$P_C := \text{conv} \left\{ y \in \mathbb{R}^{|E|} : y = \chi^{\delta(S)}, S \subseteq V \right\}.$$

If  $F = f(V)$  then MB is equivalent to the maximum cut problem (using the negative cost function) and  $P_B = P_C$ . For  $F = \lceil \frac{1}{2}f(V) \rceil$  MB is equivalent to the [\[5\]](#) and the bisection cut polytope equals the [\[45\]](#). Furthermore, MB is a special case of the minimum node capacitated graph partitioning problem (MNCGP) [\[8\]](#), where more than two clusters are available for the partition of the node set and each cluster has a common limited capacity. The objective in MNCGP is the same as in MB, i.e., to minimize the total cost of edges having endpoints in distinct clusters. Finally, we mention the [\[21\]](#)

$$P_K := \text{conv} \left\{ x \in \{0, 1\}^{|V|} : \sum_{v \in V} f_v x_v \leq F \right\},$$

which plays a fundamental role in valid inequalities for  $P_B$ . Graph partitioning problems in general have numerous applications, for instance in numerics [\[10\]](#), VLSI-design [\[18\]](#), compiler-design [\[16\]](#) and frequency assignment [\[7\]](#). A large variety of valid inequalities for the polytope associated with MNCGP is known [\[4, 8, 15\]](#) and, since MB is a special case of MNCGP, all those inequalities are also valid for  $P_B$ . A recent successful study of a combined semidefinite polyhedral branch-and-cut approach for max-cut is [\[20\]](#), it is designed for rather dense graphs with up to 400 nodes. In contrast, our semidefinite branch-and-cut approach is applicable to sparse graphs with up to 2000 nodes. In addition, we present a direct comparison with an LP approach within the same branch-and-cut environment where both approaches use the same separation routines.

In [\[2\]](#) we give a detailed analysis of  $P_B$  including several classes of new and facet-defining inequalities. We summarize these results and those from the literature in Sect. [\[2\]](#). We use these inequalities to derive and strengthen two relaxations for MB. One is based on an integer programming, the second on a semidefinite programming formulation. We develop in Sect. [\[3\]](#) both an LP-based branch-and-cut algorithm and an SDP based branch-and-cut algorithm using the same framework SCIP [\[1\]](#). In Sect. [\[4\]](#) we give a comprehensive computational comparison of both approaches on various test instances with some surprising outcomes.

## 2 Valid Inequalities for $P_B$

A large variety of valid inequalities for the cut polytope, the equipartition polytope, and the polytope associated with MNCGP is known: cycle inequalities [\[3\]](#) of the cut polytope; tree, star, and cycle inequalities [\[4\]](#) as well as suspended

tree and path block cycle inequalities [6,15] for the equipartition polytope; tree, star, cycle with ear, cycle with tails, and knapsack tree inequalities [8] valid for the polytope associated with MNCGP. Since MB is a special case of MNCGP and  $P_B \subseteq P_C$  the bisection cut polytope inherits most of the valid inequalities listed above.

For convenience we cite the cycle inequalities which we will later use in both models for MB. Let the subgraph  $C = (V_C, E_C)$  be a cycle in  $G$ . Let  $D$  be a subset of  $E_C$  such that  $|D|$  is odd. Then the

$$\sum_{e \in D} y_e - \sum_{e \in E_C \setminus D} y_e \leq |D| - 1 \tag{1}$$

is a valid inequality for the cut polytope  $P_C$ .

The cut structure implies that whenever there is a walk between two nodes of the graph with an even number of edges in the cut, the two end-nodes of the walk have to be in the same cluster. In particular, given a special root node  $r$ , a walk  $P_{rv}$  in  $G$  to some node  $v$ , an edge subset  $H_v \subseteq P_{rv}$  of even cardinality, and an incidence vector  $y$  of a cut; if the term

$$1 - \sum_{e \in P_{rv} \setminus H_v} y_e - \sum_{e \in H_v} (1 - y_e) \tag{2}$$

evaluates to one then  $r$  and  $v$  are on the same side of the cut; for nodes in opposite clusters, it is at most zero. In [2] this is used to set up an inequality linking the cut structure and the capacity constraint on the node weights.

**Proposition 1 (bisection knapsack walk inequality [2]).** Let  $\sum_{v \in V} a_v x_v \leq a_0$  be a knapsack inequality with  $a_v \geq 0, \forall v \in V$ . Let  $V' \subseteq V$  and  $r \in V'$ . Let  $P_{rv} \subseteq E$  and  $H_v \subseteq P_{rv}$  with  $|H_v|$  even. Then the bisection knapsack walk inequality

$$\sum_{v \in V'} a_v \left( 1 - \sum_{e \in P_{rv} \setminus H_v} y_e - \sum_{e \in H_v} (1 - y_e) \right) \leq a_0. \tag{3}$$

is valid for  $P_B$ .

Given a root node  $r$  and a vector  $y \in [0, 1]^{|E|}$  the optimal walks  $P_{rv}$  and subsets  $H_v$  maximizing (2) can be found in polynomial time with an algorithm that follows the one for separating cycle inequalities [2].

The inequalities (3) of [8] form a special case, where the walks  $P_{rv}$  are taken from a tree  $(T, E_T)$  of  $G$  rooted at  $r$  and  $H_v = \emptyset$  for all  $v \in V'$ ,

$$\sum_{v \in T} a_v \left( 1 - \sum_{e \in P_{rv}} y_e \right) \leq a_0. \tag{4}$$

Following [8], one may trivially strengthen the coefficients of (4) to

$$\sum_{e \in E_T} \min \left\{ \sum_{v: e \in P_{rv}} a_v, \sum_{v \in T} a_v - a_0 \right\} y_e \geq \sum_{v \in T} a_v - a_0, \tag{5}$$



we call this a *strongest truncated knapsack tree inequality*. A less obvious strengthening exploits the dependence of the coefficients in (5) on the choice of the root node, which we express by the notation

$$\alpha_0 := \sum_{v \in T} a_v - a_0, \quad \alpha_e^r := \min \left\{ \sum_{v: e \in P_{rv}} a_v, \alpha_0 \right\}, e \in E_T, \quad (6)$$

The strongest form is achieved if  $r$  enforces a sort of balance with respect to the cumulated node weights on the paths to  $r$ .

**Theorem 2.** [2] Let  $G = (T, E_T)$  be a tree with  $\sum_{v \in V} a_v x_v \leq a_0$  and let  $r \in \mathcal{R}$  :=  $\text{Argmin}_{v \in T} \sum_{e \in E_T} \alpha_e^v$ . Then  $\sum_{e \in E_T} \alpha_e^r y_e \geq \sum_{e \in E_T} \alpha_e^r y_e \geq \alpha_0$  is a valid inequality for  $P_B$ . If  $\sum_{e \in E_T} \alpha_e^r y_e = \sum_{e \in E_T} \alpha_e^r y_e$  for  $r, s \in \mathcal{R}$  and  $y \in P_B$

The elements of the set  $\mathcal{R}$  are called *minimal roots* of a given tree  $(T, E_T)$ , and by Theorem 2 all minimal roots of  $(T, E_T)$  deliver the same strongest truncated knapsack tree inequality. Additional structural results allow to locate minimal roots algorithmically at almost no cost. This strengthening proved highly effective in our experiments. Note, if the inequality induces a facet then  $r$  is a minimal root by Theorem 2. In some cases the minimal root condition is also sufficient. In order to state this result, call a path in  $(T, E_T)$  *branch-less*, if its inner nodes are all of degree 2 in the tree.

**Theorem 3.** [2] Let  $G = (T, E_T)$  be a tree with  $\sum_{v \in V} a_v x_v \leq a_0$  and let  $r \in T$  with  $f_v = 1$  for  $v \in T$  and  $\frac{|T|}{2} + 1 \leq F < |T|$ . Then  $\sum_{e \in E} \min \{ |v : e \in P_{rv}|, |V| - F \} y_e \geq |V| - F$  is a valid inequality for  $P_B$ .

- (a)  $r$  is a minimal root of  $(T, E_T)$  and  $F$  is the length of a branch-less path condition.
- (b)  $F = |T| - 1$

To motivate a strengthening for general bisection knapsack walk inequalities consider the case of a disconnected graph with two components, one of them being a single edge  $\{u, v\}$ , the other connected one being  $V' = V \setminus \{u, v\}$ . If  $y_{uv} = 1$  then  $u$  and  $v$  belong to different clusters and therefore the capacity remaining for the clusters in  $V'$  (e.g. the right-hand side of (3)) can be reduced to  $F - \min \{f_u, f_v\} y_{uv}$ . To generalize this idea we define for  $\bar{G} \subseteq G$  with  $\bar{V} \subseteq V$ ,  $\bar{E} \subseteq E(\bar{V})$  and  $a \in \mathbb{R}_+^{|\bar{V}|}$  a function  $\beta_{\bar{G}} : \{0, 1\}^{|\bar{E}|} \rightarrow \mathbb{R} \cup \infty$  with

$$\beta_{\bar{G}}(y) = \inf \left\{ a(S), a(\bar{V} \setminus S) : S \subseteq \bar{V}, \max \{ a(S), a(\bar{V} \setminus S) \} \leq a_0, y = \chi^{\delta_{\bar{G}}(S)} \right\}.$$

Now we look at the convex envelope  $\check{\beta}_{\bar{G}} : \mathbb{R}^{|\bar{E}|} \rightarrow \mathbb{R} \cup \infty$  of  $\beta_{\bar{G}}(y)$ , i.e.,

$$\check{\beta}_{\bar{G}}(y) = \sup \left\{ \check{\beta}(y) : \check{\beta} : \mathbb{R}^{|\bar{E}|} \rightarrow \mathbb{R}, \check{\beta} \text{ convex}, \check{\beta}(z) \leq \beta_{\bar{G}}(z) \text{ for } z \in \{0, 1\}^{|\bar{E}|} \right\}.$$

Note that  $\check{\beta}_{\bar{G}}$  is a piecewise linear function on its domain.

**Proposition 4 (capacity reduced bisection knapsack walk inequality [2]).** Let  $\sum_{v \in V} a_v x_v \leq a_0$ ,  $a_v \geq 0$ ,  $v \in V$ . Let  $\{V_l, E_l\}_{l=1, \dots, L}$  be a partition of  $(V, E)$  into  $L$  subgraphs  $G_l = (V_l, E_l) = G \cap (V_l, E_l)$  with  $V_l \cap V_0 = \emptyset$ ,  $E_l \subseteq E(V_l)$ ,  $l = 1, \dots, L$ . Let  $\beta_{G_l}^l$  be the capacity reduced bisection knapsack walk inequality for  $G_l$ .

$$c_0^l + \sum_{e \in E_l} c_e y_e \leq \check{\beta}_{G_l}(y) \tag{7}$$

Let  $y \in P_B$ . The capacity reduced bisection knapsack walk inequality

$$\sum_{v \in V_0} a_v \left( 1 - \sum_{e \in P_{rv} \setminus H_v} y_e - \sum_{e \in P_{rv} \cap H_v} (1 - y_e) \right) \leq a_0 - \sum_{l=1}^L (c_0^l + \sum_{e \in E_l} c_e y_e) \tag{8}$$

is in  $P_B$ .

In certain cases it is possible to establish a full description of  $\check{\beta}_{\bar{G}}$  via a complete description of the cluster weight polytope defined as follows. Given a graph  $G = (V, E)$  with non-negative node weights  $a_v \in \mathbb{R}$  for all  $v \in V$ . For a set  $S \subseteq V$  we define  $h(S) := (a(S), (\chi^{\delta(S)})^T)^T \in \mathbb{R}^{|E|+1}$ . With respect to a given  $a_0 \in \mathbb{R}$  we call

$$P_{CW} = \text{conv} \{ h(S) : S \subseteq V, a(S) \leq a_0, a(V \setminus S) \leq a_0 \}$$

the cluster weight polytope.

In [2], a full description of  $P_{CW}(\bar{G})$  is given for the special case that the subgraph  $\bar{G} = (\bar{V}, \bar{E})$  is a star centered at some node  $r \in \bar{V}$ ,  $a \geq 0$ , and  $a_0$  satisfies  $a(\bar{V}) \leq a_0$ . In order to state the nontrivial facets of this case, assume  $a(\bar{V} \setminus \{r\}) > a_r$  and call a triple  $(V_p, \bar{v}, V_n)$  feasible if it fulfills  $\bar{V} = \{r, \bar{v}\} \cup V_p \cup V_n$  and  $a(V_p) \leq \frac{1}{2}a(\bar{V}) < a(V_p) + a_{\bar{v}}$ . For all feasible triples  $(V_p, \bar{v}, V_n)$  the inequalities

$$y_0 - \sum_{v \in V_p} a_v y_{rv} - (a(\bar{V}) - 2a(V_p) - a_{\bar{v}}) y_{r\bar{v}} + \sum_{v \in V_n} a_v y_{rv} \geq 0 \tag{9}$$

are the facet-inducing inequalities for  $P_{CW}(\bar{G})$ . Thus, inequalities (9) form the best linear minorants (7) to be used in (8) in this case.

In our experiments this rather involved strengthening technique proved far less effective than the simple root strengthening for knapsack tree inequalities, but this may be due to the dominating non-negative cost structure in our experiments.

### 3 Linear and Semidefinite Relaxations for MB

The linear relaxation for MB is derived from the following integer linear programming formulation. We select a node  $s \in V$  and extend  $E$  so that  $s$  is adjacent to

all other nodes in  $V$ , setting the weights  $w(\cdot)$  of new edges to zero. We introduce binary variables  $y_{ij}$  for all  $ij \in E$  and require that  $y_{ij} = 1$  if nodes  $i$  and  $j$  are in different clusters and  $y_{ij} = 0$  otherwise. The capacity constraints on the two clusters can then be formulated as

$$f_s + \sum_{i \in V \setminus \{s\}} f_i(1 - y_{is}) \leq F, \tag{10}$$

$$\sum_{i \in V \setminus \{s\}} f_i y_{is} \leq F. \tag{11}$$

Thus we obtain the following integer linear model for MB.

$$\begin{aligned} \min \quad & \sum_{e \in E} w_e y_e \\ \text{s.t.} \quad & \text{(I)}, \text{(II)}, \text{(III)}, \\ & y \in \{0, 1\}^{|E|}. \end{aligned} \tag{12}$$

The cycle inequalities (I) make sure that each solution to (II) corresponds to an incidence vector of a cut in  $G$ . (II) and (III) require that this cut is a bisection cut. Although the number of all valid cycle inequalities for  $P_B$  is exponential in  $|E|$ , the inequalities can be separated in polynomial time [3].

Our semidefinite relaxation for MB follows the classical approach of [19]. Given the weighted adjacency matrix  $W$  of  $G$ , represent a bipartition by  $x \in \{-1, 1\}^n$  with  $x_i = -1$  if  $x \in S$  and  $x_i = 1$  if  $x \in V \setminus S$ . Then an integer quadratic model for MB reads

$$\min \left\{ \sum_{i < j} w_{ij} \frac{1 - x_i x_j}{2} : |f^T x| \leq 2F - f(V), x \in \{-1, 1\}^{|V|} \right\}. \tag{13}$$

Rewrite  $\sum_{i < j} w_{ij} \frac{1 - x_i x_j}{2}$  as  $\langle \frac{1}{4}L, xx^T \rangle$ , where  $L = \text{Diag}(We) - W$  is the weighted Laplace matrix of  $G$ , and relax  $xx^T$  for  $x \in \{-1, 1\}^n$  to  $X \succeq 0$  with  $\text{diag}(X) = e$  to obtain

$$\begin{aligned} \min \quad & \langle \frac{1}{4}L, X \rangle \\ \text{s.t.} \quad & \langle ff^T, X \rangle \leq (2F - f(V))^2, \\ & \text{diag}(X) = e, \\ & X \succeq 0. \end{aligned} \tag{14}$$

The framework employs the same separation algorithms for (I3) and (I4) by transforming a  $\bar{X}$  to a  $\bar{y}$  via  $\bar{y}_{ij} = \frac{1 - \bar{X}_{ij}}{2}$  for all  $\{i, j\} \in E$ . Separated inequalities  $\sum_{\{i, j\} \in E} k_{ij} y_{ij} \leq k_l$  are translated into constraints  $\langle K, X \rangle \leq k_s$  for the primal semidefinite relaxation by  $K_{ij} = -\frac{1}{2}k_{ij}$  for all  $\{i, j\} \in E$  and  $k_s = 2k_l - \sum_{\{i, j\} \in E} k_{ij}$ .

Our branch-and-cut implementation using the linear relaxation follows basically standard techniques known in the community. Our implementation with

the semidefinite relaxation is, however, not straightforward and involves many details of which we sketch a few. In contrast to [20], where a standard polyhedral bundle method is used together with a rather expensive semidefinite oracle, we solve the semidefinite relaxation approximately by applying the spectral bundle method [14,13] to the dual of (14) in its equivalent form as an eigenvalue optimization problem,

$$- \min_{\substack{z \in \mathbb{R}^{|V|} \\ p \geq 0}} |V| \lambda_{\max} \left( -\frac{1}{4}L + \text{Diag}(z) - ff^T p \right) - \langle e, z \rangle + (2F - f(V))^2 p. \quad (15)$$

In this setting, the oracle is a Lanczos method for computing extremal eigenvalues and corresponding eigenvectors of large structured matrices; we use the eigenvectors to form a semidefinite cutting model. Any dual feasible solution of (15) yields a valid lower bound for MB. The decisive step in the use of the spectral bundle method in branch-and-cut is to exploit its restarting properties and its primal approximate solution.

While solving (15) the bundle method aggregates the eigenvector information to an approximate primal solution  $\tilde{X}$  of (14) of the form

$$\tilde{X} = PUP^T + \alpha \overline{W} \succeq 0, \quad (16)$$

where  $P \in \mathbb{R}^{n \times k}$ ,  $P^T P = I$ , holds a basis of the aggregated eigenvectors,  $U \in S_+^k$ ,  $\alpha \geq 0$  so that  $\text{tr} U + \alpha = |V|$ , and  $\overline{W} \in S_+^n$  is sparse with  $\text{tr} \overline{W} = 1$ . The software allows to choose the support of  $\overline{W}$ . We start with the support of  $L$  and extend it on the fly by further off-diagonal elements (edges) that promise to be useful in the separation of cycle inequalities; this proved to be highly effective already in [12]. The approximate solution  $\tilde{X}$  will in general satisfy the constraints approximately only. On the one hand this may result in off-diagonal elements  $\tilde{X}_{ij}$  outside of the interval  $[-1, 1]$ , so  $\tilde{X}$  has to be rounded or truncated before the standard separation routines can be applied (see above for the transformation to  $y$ ). On the other hand, a separated inequality may still be violated after the next optimization run, so precautions have to be taken against separating the same inequality again and again. Such aspects have been addressed in [12] and we build on this work.

For generating primal solutions we use heuristics, similar in style to Goemans-Williamson [11], on the spectral bundle part  $PUP^T$  of the approximate primal solution  $\tilde{X}$ . One of the more successful variants pays special attention to the sign structure of the large eigenvalues. We improve these rounded solutions by simple local search techniques. The good quality of these solutions proved to be one major advantage of SDP over LP in our branch-and-cut comparisons.

After the addition of newly separated cutting planes there is no difficulty in restarting the bundle method from the old Lagrange multiplier solution by setting the new multipliers to zero. Extending the old subgradients to the new coordinates can be done easily, if the support of the new inequalities is restricted to the support of  $\overline{W}$ . This way the bundle model needs not be rebuilt in spite of the changes in dimension. Fortunately, no dramatic scaling problems seem

to arise during the usual separation process, maybe because violation of the inequalities and changes in the multipliers seem to converge to zero at a common speed. Quite often, however, we observed significant scaling problems when the relaxation needs to be resolved after the addition of a branching constraint like setting  $X_{ij} = 1$ . Indeed, a few of the Lagrange multipliers – those associated with the new constraint  $X_{ij} = 1$  and with the constraints containing the newly restricted edge – will typically change a lot, but most other multipliers seem not to move much. In this context the following idea for a scaling heuristic turned out to be quite effective. Take the two eigenvectors  $v, w$  to the two nonzero eigenvalues of  $X_{ij}$  and allow the more or the less change in the Lagrange multipliers in dependence on the Ritz values  $v^T A v$  and  $w^T A w$  of each constraint matrix  $A$ .

In a combined bundle and cutting plane approach a heavy tailing off effect has to be expected and can be observed in solving the relaxation. To cope with this, we never wait for convergence but stop solving the relaxation quite early on several “lack of progress” criteria. Yet, the resulting rough approximations still need quite some time. Consequently the number of branch-and-bound nodes stays small and the common strong branching techniques of LP cannot be applied. To make up for this we developed a rather elaborate branching rule. Based on the vector labelling corresponding to  $PUP^T$  we try to cluster the nodes into subsets having well aligned vectors and investigate the effect on the cost function if two such clusters are forced to be aligned in the same or in the opposite direction. Given the two clusters where this shows the strongest effect, we pick a representative of each cluster, say nodes  $\hat{i}$  and  $\hat{j}$ , and set  $X_{ij}$  to  $+1$  in one subproblem and to  $-1$  in the other.

## 4 Computational Results

For our empirical investigations we used sparse graph instances from the samples presented in [17] varying in the number of edges between 1 500 and 500 000. We generally set  $F = 0.525f(V)$ . As a branch-and-cut framework we use SCIP [1] with ILOG CPLEX 9.130 as LP-solver and the spectral bundle method developed by [13] as the SDP-solver. The computations were executed on a 3.2 GHz Pentium IV processor with 1 GB RAM.

In Table 1 we present a comparison of the performance of the cutting plane algorithms based on the knapsack tree inequalities with minimal roots (1), capacity improved bisection knapsack walk inequalities (2) and cycle inequalities (3) in combination with the linear and the semidefinite relaxation. Note that the cycle inequalities are part of the integer linear model (12), so they are separated in each setting of the LP relaxation. In the tests of Table 1 we only computed the root node of the branch-and-cut tree. We added inequalities of each class separately as long as violated cuts were found, the time limit of 4 hours was not exceeded and a heuristically computed upper bound was not yet proven to be optimal. Along with the lower bounds we report the best upper bounds known to us but not necessarily achieved in the same computations.

**Table 1.** Lower bounds computed by the linear and the semidefinite relaxation in the root of the b&b tree. Results on VLSI design graphs [17].

graph $n.m$	linear relaxation			semidefinite relaxation					$ub$	
	$cy$	$cy+bkw$	$cy+kt$	$all$	$none$	$bkw$	$kt$	$cy$		$all$
diw681.1494	18.4	134.9	<b>136.3</b>	135.9	77.3	135.1	134.7	<b>140.6</b>	137.9	142
taq1021.2253	23.1	74.1	113.2	<b>113.9</b>	60.1	112.9	112.4	<b>116.8</b>	115.0	118
dmxal755.3686	0.0	42.6	87.1	<b>91.1</b>	37.5	89.3	89.0	<b>92.9</b>	90.3	94
diw681.3104	34.8	238.4	744.4	<b>829.2</b>	630.7	954.6	935.0	<b>988.8</b>	969.4	1011
taq334.3763	75.5	111.8	324.8	<b>324.9</b>	234.0	<b>324.8</b>	320.5	317.9	324.7	342
diw681.6402	46.8	136.2	304.6	<b>306.3</b>	280.8	320.7	310.4	319.7	<b>322.7</b>	331
gap2669.6182	8.6	28.3	71.1	<b>73.7</b>	35.8	72.7	<b>74.0</b>	<b>74.0</b>	73.0	74
alut2292.6329	3.8	17.1	55.3	<b>59.8</b>	39.7	74.0	74.6	<b>76.2</b>	74.9	77
taq1021.5480	74.1	154.4	639.8	<b>689.9</b>	1122.0	1510.8	1469.9	<b>1540.6</b>	1538.9	1650
dmxal755.10867	20.4	31.7	137.1	<b>138.6</b>	94.6	143.0	142.0	143.5	<b>143.7</b>	150
alue6112.16896	0.0	8.2	7.8	<b>31.0</b>	52.9	117.6	99.9	<b>135.1</b>	98.0	136
gap2669.24859	<b>55.0</b>	55.0	55.0	55.0	46.0	55.0	55.0	55.0	55.0	55
taq1021.31641	151.8	215.1	372.5	<b>374.6</b>	359.4	386.6	301.8	<b>398.6</b>	396.7	404
alut2292.494500	559.0	740.3	1571.3	<b>1966.2</b>	<b>5395.0</b>	53374	46405	51071	47007	67815
mean lower bd	23	77	162	<b>186</b>	184	282	270	<b>289</b>	279	
mean time	40	8389	11646	10267	120	5684	3386	1476	5684	

$n$  number of nodes,  $m$  number of edges,  $ub$  upper bound

**Table 2.** Performance of the linear and the semidefinite branch-and-cut algorithms. Results on VLSI design graphs [17].

graph $n,m$	linear relaxation				semidefinite relaxation					
	# $b\mathcal{E}b$ modes	time	upper bound	lower bound	gap (%)	# $b\mathcal{E}b$ modes	time	upper bound	lower bound	gap (%)
diw681.1494	1686	10h	144	140.8	2	237	10h	142	140.5	1
taq1021.2253	95	10h	118	118.0	0	1	322s	118	118.0	0
dmxal755.3686	35	9h	94	94	0	68	10h	94	92.8	1
diw681.3104	1	10h	1064	835.7	27	124	10h	1011	1007.1	0.1
taq334.3763	351	4h	342	342.0	0	2318	10h	342	340.1	0.4
diw681.6402	3	10h	357	315.2	13	159	10h	331	329.2	0.1
gap2669.6182	1	4h	74	74	0	1	651s	74	74	0
alut2292.6329	1	10h	77	69.5	10	96	10h	77	76.1	1
taq1021.5480	1	10h	2019	701.2	187	84	10h	1650	1586.9	3
dmxal755.10867	62	10h	157	144.1	8	79	10h	150	145.9	2
alut6112.16896	1	10h	146	21.5	578	11	10h	136	135.6	0
gap2669.24859	1	2525s	55	55.0	0.0	1	491s	55	55.0	0
taq1021.31641	1	10h	426	375.3	13	9	10h	404	399.0	1
alut2292.494500	1	*5h	67815	1813.5	3639	1	5h	67815	51880.0	30
geom. mean**	12	27144s	208	156	8	42	17278s	199	197	1

\* Early termination due to a memory shortage, \*\* alut2292.494500 excluded.

Within the linear relaxation, the knapsack tree inequalities have the biggest impact on the improvement of the lower bound. This may seem surprising in view of the fact that the knapsack walk inequalities subsume the knapsack tree inequalities; the reason is that, for speed, both separators start from a few seed nodes and then the minimal root strengthening of Theorem 2 boosts the performance of knapsack trees. Column 5 shows that it is worth to apply all separators in the linear case. The cycle separator alone achieves very poor lower bounds, so studying the bisection cut polytope  $P_B$  pays off when considering the linear relaxation of MB. The situation is stunningly different in the semidefinite case. Here, the pure semidefinite relaxation (11) yields already good lower bounds. For very large instances like  $n = 1000$  the separation routines only slow down the solution process and thus lead to worse bounds when the computing time is a major limiting factor. The best results are achieved when the separator for cycle inequalities is used exclusively. The bisection specific inequalities, i.e., the knapsack tree and bisection knapsack walk inequalities, yield roughly the same performance. These also improve the bound significantly but in comparison to cycle inequalities computation times are higher.

**Table 3.** Performance of the linear and the semidefinite branch-and-cut algorithms. Results on compiler design graphs [16].

graph $n.m$	linear relaxation					semidefinite relaxation				
	# b&B nodes	time (sec.)	upper bound	lower bound	gap (%)	# b&B nodes	time (sec.)	upper bound	lower bound	gap (%)
cb.30.47	354	1	266	266	0	25166	540	266	266	0
cb.30.56	326	2	379	379	0	10276	256	379	379	0
cb.45.98	49	5	989	989	0	2995	438	989	989	0
cb.47.101	100	3	527	527	0	4403	433	527	527	0
cb.47.99	12	5	765	765	0	963	113	765	765	0
cb.61.187	785	81	2826	2826	0	10333	*5907	2826	2647	7

\* Early termination due to a memory shortage.

Based on the results of Table 1 the parameters of our branch-and-cut codes were set as follows. For the LP-relaxation the knapsack tree separator is given the highest priority and separation frequency, followed by the cycle and the bisection knapsack walk separators. For the semidefinite relaxation the cycle separator is the only separator. We limited computation time to 10 hours for each instance. The computations are presented in Tables 2 and 3. By solving big instances (Table 2) the SDP-relaxation outperforms the LP-relaxation with respect to both quality of dual bounds and computation time. In most cases we also observed that after a few seconds the value of the current SDP-bound exceeds the value of the current LP-bound and stays ahead throughout. However, the situation



looks quite different when solving graph instances coming from compiler design as in [16]. For these instances the linear solver is far ahead of the semidefinite one with respect to computation time.

## 5 Conclusion

In this paper we considered the minimum graph bisection problem, a combinatorial problem for which linear and semidefinite relaxations are easy to derive. Using previous polyhedral studies presented in [2] we developed separation routines for valid inequalities to the bisection cut polytope  $P_B$  and incorporated them into a common branch-and-cut framework for linear and semidefinite relaxations. On the basis of large sparse instances coming from VLSI design we showed the good performance of the semidefinite approach versus the mainstream linear one.

**Acknowledgments.** This work was supported by the German Research Foundation (DFG) under grants HE 3524/2-1/2 and MA 1324/2-1/2.

## References

1. Achterberg, T.: Constraint integer programming. PhD-Thesis, Technische Universität Berlin, Berlin (2007)
2. Armbruster, M., Fügenschuh, M., Helmberg, C., Martin, A.: On the bisection cut polytope. Technical Report, Chemnitz/Darmstadt University of Technology (2007)
3. Barahona, F., Mahjoub, A.R.: On the cut polytope. *Math. Prog.* 36, 157–173 (1986)
4. Conforti, M., Rao, M.R., Sassano, A.: The equipartition polytope I, II. *Math. Prog.* 49, 49–70 (1990)
5. de Souza, C.C.: The graph equipartition problem: Optimal solutions, extensions and applications. PhD-Thesis, Université Catholique de Louvain, Belgium (1993)
6. Deza, M., Laurent, M.: *Geometry of Cuts and Metrics Algorithms and Combinatorics*, vol. 15. Springer, Heidelberg (1997)
7. Eisenblätter, A.: Frequency Assignment in GSM Networks. PhD-Thesis, Technische Universität Berlin, Berlin (2001)
8. Ferreira, C.E., Martin, A., de Souza, C.C., Weismantel, R., Wolsey, L.A.: Formulations and valid inequalities for the node capacitated graph partitioning problem. *Math. Prog.* 74, 247–266 (1996)
9. Garey, M.R., Johnson, D.S.: *Computers and Intractability*. W.H. Freeman and Company, New York (1979)
10. Gilbert, J.R., Tarjan, R.E.: The analysis of a nested dissection algorithm. *Numer. Math.* 50, 377–404 (1979)
11. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* 42, 1115–1145 (1995)
12. Helmberg, C.: A cutting plane algorithm for large scale semidefinite relaxations. In: Grötschel, M. (ed.) *The Sharpest Cut*. MPS-SIAM Series on Optimization, pp. 233–256 (2004)
13. Helmberg, C., Kiwiel, K.C.: A Spectral Bundle Method with Bounds. *Math. Prog.* 93, 173–194 (2002)

14. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. *SIAM J. Optim.* 10(3), 673–696 (2000)
15. Laurent, M., de Souza, C.C.: Some new classes of facets for the equicut polytope. *Discr. App. Math.* 62, 167–191 (1995)
16. Johnson, E., Mehrotra, A., Nemhauser, G.: Min-cut clustering. *Math. Prog.* 62, 133–152 (1993)
17. Jünger, M., Martin, A., Reinelt, G., Weismantel, R.: Quadratic 0/1 optimization and a decomposition approach for the placement of electronic circuits. *Math. Prog. B* 63(3), 257–279 (1994)
18. Lengauer, T.: *Combinatorial algorithms for integrated circuit layout*. John Wiley and Sons Ltd., Chichester (1990)
19. Poljak, S., Rendl, F.: Nonpolyhedral relaxations of graph-bisection problems. *SIAM J. Optim.* 5(3), 467–487 (1995)
20. Rendl, F., Rinaldi, G., Wiegele, A.: A branch and bound algorithm for Max-Cut based on combining semidefinite and polyhedral relaxations. In: Fischetti, M., Williamson, D.P. (eds.) *IPCO 2007*. LNCS, vol. 4513, pp. 295–309. Springer, Heidelberg (2007)
21. Weismantel, R.: On the 0/1 Knapsack polytope. *Math. Prog.* 77, 49–68 (1997)

# Binary Positive Semidefinite Matrices and Associated Integer Polytopes

Adam N. Letchford<sup>1,\*</sup> and Michael M. Sørensen<sup>2</sup>

<sup>1</sup> Department of Management Science, Lancaster University,  
Lancaster LA1 4YW, United Kingdom

A.N.Letchford@lancaster.ac.uk

<sup>2</sup> CORAL & Department of Business Studies, Aarhus School of Business,  
University of Aarhus, Denmark

mim@asb.dk

**Abstract.** We consider the positive semidefinite (psd) matrices with binary entries. We give a characterisation of such matrices, along with a graphical representation. We then move on to consider the associated integer polytopes. Several important and well-known integer polytopes — the cut, boolean quadric, multicut and clique partitioning polytopes — are shown to arise as projections of binary psd polytopes. Finally, we present various valid inequalities for binary psd polytopes, and show how they relate to inequalities known for the simpler polytopes mentioned. Along the way, we answer an open question in the literature on the max-cut problem, by showing that the so-called *k-gonal* inequalities define a polytope.

**Keywords:** Polyhedral combinatorics, semidefinite programming.

## 1 Introduction

A real square symmetric matrix  $M \in \mathbb{R}^{n \times n}$  is said to be (psd) if and only if any of the following (equivalent) conditions hold:

- $a^T M a \geq 0$  for all  $a \in \mathbb{R}^n$ ,
- all principal submatrices of  $M$  have non-negative determinants,
- there exists a real matrix  $A$  such that  $M = A A^T$ .

The set of psd matrices of order  $n$  forms a convex cone in  $\mathbb{R}^{n \times n}$  (e.g., Hill & Waters [14]), and is often denoted by  $\mathcal{S}_+^n$ .

In this paper, we consider the  $\{0, 1\}^{n \times n}$  psd matrices, i.e., psd matrices belonging to  $\{0, 1\}^{n \times n}$ , and the associated family of integer polytopes, which we call  $\mathcal{P}_M$ . Although psd matrices and semidefinite programming have received much interest from the integer programming and combinatorial optimisation community (see the surveys Goemans [11] and Laurent & Rendl

---

\* The research of the first author was supported by the Engineering and Physical Sciences Research Council under grant EP/D072662/1.

[17]), these specific matrices and polytopes appear to have received no attention. This is remarkable, because, as we will see, the binary psd matrices can be easily characterised, and they have a natural graphical interpretation. Moreover, several important and well-known integer polytopes — such as the  $\{0, 1\}^n$ ,  $\{0, 1\}^n$  and  $\{0, 1\}^n$  polytopes — can in fact be viewed as nothing but faces of binary psd polytopes. In that sense, the binary psd polytopes form an important, and hitherto overlooked, family of ‘master’ polytopes for combinatorial optimisation.

The paper is structured as follows. In Sect. 2 we characterise the binary psd matrices and show how this leads to the graphical representation. In Sect. 3 we formally define the binary psd polytopes and show how they are related to the other polytopes mentioned. Finally, in Sect. 4 we present several classes of valid and facet-inducing linear inequalities for binary psd polytopes, and show how they imply many of the known inequalities for the simpler polytopes mentioned above. As a by-product of our analysis, we obtain (in Subsect. 4.2) a remarkably simple proof that the so-called  $k$ -cut inequalities define a polytope — thus settling an open question in the literature on the max-cut problem (see Avis & Umemoto [2]).

## 2 Characterisation

We now give our first characterisation of the binary psd matrices. Note that the symmetric rank one binary matrices are those that can be written in the form  $vv^T$  for some binary vector  $v \in \{0, 1\}^n$ .

**Proposition 1.** *A symmetric matrix  $M \in \mathbb{R}^{n \times n}$  is a binary psd matrix if and only if  $M_{ii} \in \{0, 1\}$  for all  $i \in \{1, \dots, n\}$  and  $M_{ij} = M_{ji} = 0$  for all  $i, j \in \{1, \dots, n\}$  with  $i \neq j$ .*

The ‘if’ part follows trivially from the fact that  $S_+^n$  is a cone. We prove the ‘only if’ part. Suppose that  $M$  is a binary psd matrix. Since all  $2 \times 2$  principal submatrices of  $M$  must have non-negative determinant, we have that, if  $M_{ii} = 0$  for some  $i \in \{1, \dots, n\}$ , then  $M_{ij} = M_{ji} = 0$  for  $j = 1, \dots, n$ . Thus, if we let  $R = \{i \in \{1, \dots, n\} : M_{ii} = 1\}$ , we have that  $M$  has zero entries outside the principal submatrix defined by the row/column indices in  $R$ . This submatrix, which must also be psd, has 1s on the main diagonal. The fact that a symmetric binary matrix with 1s on the main diagonal is psd if and only if it is the sum of symmetric rank one binary matrices is well-known and easy to prove: see, e.g., Lemma 1 of Dukanovic & Rendl [10].  $\square$

We note in passing the following corollary:

**Corollary 1.** *A symmetric matrix  $M \in \mathbb{R}^{n \times n}$  is a binary psd matrix if and only if  $M_{ii} \in \{0, 1\}$  for all  $i \in \{1, \dots, n\}$  and  $M_{ij} = M_{ji} = 0$  for all  $i, j \in \{1, \dots, n\}$  with  $i \neq j$ .*

The ‘if’ part follows immediately from the definitions. We show the ‘only if’ part. Let  $M \in \{0, 1\}^{n \times n}$  be a binary psd matrix. If  $M$  is the zero matrix, the

result is trivial. Otherwise, from Proposition 1, there exists a positive integer  $p$  and vectors  $v^1, \dots, v^p \in \{0, 1\}^n$  such that:

$$M = \sum_{k=1}^p v^k (v^k)^T.$$

If we let  $A$  be the  $n \times p$  matrix whose  $k$ th column is the vector  $v^k$ , we have that  $M = AA^T$ . Thus,  $M$  is completely positive.  $\square$

The following proposition gives an alternative characterisation of the binary psd matrices, in terms of linear inequalities:

**Proposition 2.** *Let  $M \in \{0, 1\}^{n \times n}$  with  $n \geq 3$ . Then  $M$  is binary psd if and only if*

$$M_{ij} \leq M_{ii} \quad (1 \leq i < j \leq n) \tag{1}$$

$$M_{ik} + M_{jk} \leq M_{kk} + M_{ij} \quad (1 \leq i < j \leq n; k \neq i, j). \tag{2}$$

It is easy to check that the inequalities (1) and (2) are satisfied by symmetric rank one binary matrices. Proposition 1 then implies that they are satisfied by binary psd matrices (since both sets of inequalities are homogeneous). Now, suppose that a symmetric binary matrix  $M$  satisfies the inequalities (1) and (2). If  $M_{ii} = 0$  for a given  $i$ , the inequalities (1) imply that  $M_{ij} = M_{ji} = 0$  for all  $j \neq i$ . Thus, just as in the proof of Proposition 1, we can assume that  $M$  has 1s on the main diagonal. Now note that, if  $M_{ik} = M_{jk} = 1$  for some indices  $i, j, k$ , then the inequalities (2) ensure that  $M_{ij} = 1$ . By transitivity, this implies that  $\{1, \dots, n\}$  can be partitioned into subsets in such a way that, for all pairs  $i, j$ ,  $M_{ij} = 1$  if and only if  $i$  and  $j$  belong to the same subset. That is to say,  $M$  is the sum of one or more symmetric rank one binary matrices. By Proposition 1,  $M$  is psd.  $\square$

We will also find the following simple result useful later on (see Proposition 6 in Subsect. 3.1):

**Proposition 3.** *Let  $M \in \{0, 1\}^{n \times n}$  with  $M_{rr} = 0$  for  $1 \leq r \leq n$ . Then  $M$  is binary psd if and only if*

If  $M$  satisfies the inequalities (1) and (2), then the modified matrix will also satisfy them.  $\square$

Finally, we point out that the binary psd matrices have a natural graphical representation. Given an  $n \times n$  binary psd matrix  $M$ , we construct a subgraph of the complete graph  $K_n$  as follows. The vertex  $i$  is included in the subgraph if and only if  $M_{ii} = 1$ , and the edge  $\{i, j\}$  is included if and only if  $M_{ij} = 1$ . The symmetric rank one binary matrices then correspond to cliques in  $K_n$ , if we define cliques in a slightly non-conventional way, so that they consist, not only of vertices, but also of the edges between them. The binary psd matrices correspond to unions of node-disjoint cliques.

### 3 Polytopes

In this section, we formally define the binary psd polytope and show how it is related to certain other polytopes in combinatorial optimisation.

#### 3.1 The Binary Psd Polytope

Note that any binary psd matrix  $M$ , being symmetric, satisfies the  $\binom{n}{2}$  equations  $M_{ij} = M_{ji}$  for all  $1 \leq i < j \leq n$ . Therefore, if we defined the binary psd polytope in  $\mathbb{R}^{n \times n}$ , it would not be full-dimensional. Therefore, we decided to work in  $\mathbb{R}^{\binom{n+1}{2}}$  instead.

We will need the following notation. We let  $V_n = \{1, \dots, n\}$  and  $E_n = \{S \subset V_n : |S| = 2\}$ . We then define, for all  $i \in V_n$ , the binary variable  $x_i$ , which takes the value 1 if and only if  $M_{ii} = 1$ ; and we define, for all  $\{i, j\} \in E_n$ , the binary variable  $y_{ij}$ , which takes the value 1 if and only if  $M_{ij} = M_{ji} = 1$ . We denote by  $\mathcal{M}(x, y)$  the linear operator that maps a given pair  $(x, y) \in \{0, 1\}^{V_n \cup E_n}$  onto the corresponding  $n \times n$  symmetric matrix. Then, the  $\mathcal{P}_n$  of order  $n$  is defined as:

$$\mathcal{P}_n = \text{conv} \{ (x, y) \in \{0, 1\}^{V_n \cup E_n} : \mathcal{M}(x, y) \in \mathcal{S}_+^n \}.$$

For  $n = 2$ , there are 5 binary psd matrices:

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

The corresponding vectors  $(x_1, x_2, y_{12})$  are  $(0, 0, 0)$ ,  $(1, 0, 0)$ ,  $(0, 1, 0)$ ,  $(1, 1, 0)$  and  $(1, 1, 1)$ , respectively. The polytope  $\mathcal{P}_2$  is described by the linear inequalities  $x_1 \leq 1$ ,  $x_2 \leq 1$ ,  $y_{12} \geq 0$ ,  $y_{12} \leq x_1$  and  $y_{12} \leq x_2$ .

Proposition 2 enables us to define  $\mathcal{P}_n$  more explicitly.

**Proposition 4.** For  $n \geq 3$ ,  $\mathcal{P}_n = \{ (x, y) \in \{0, 1\}^{V_n \cup E_n} :$

$$y_{ij} \leq x_i \quad (i \in V_n, j \in V_n \setminus \{i\}) \tag{3}$$

$$y_{ik} + y_{jk} \leq x_k + y_{ij} \quad (\{i, j\} \in E_n, k \in V_n \setminus \{i, j\}). \tag{4}$$

The following result is also easy to prove:

**Proposition 5.** For  $n \geq 3$ ,  $\mathcal{P}_n = \{ (x, y) \in \{0, 1\}^{V_n \cup E_n} :$

$$x_i \leq 1 \quad i \in V_n$$

$$y_e \geq 0 \quad e \in E_n$$

and the inequalities (3) and (4). □ □

Finally, Proposition 3 can be used to show the following result:

**Proposition 6.** Let  $\mathcal{P}_n$  be the polytope defined by  $x_i \leq 1$ ,  $i \in V_n$ ,  $b^T y \leq a^T x + c$ ,  $a \geq 0$ ,  $c \geq 0$ .

### 3.2 The Boolean Quadric Polytope

The Boolean quadric polytope (Padberg [21]) of order  $n$  is defined as:

$$\text{BQP}_n = \text{conv} \{ (x, y) \in \{0, 1\}^{V_n \cup E_n} : y_{ij} = x_i x_j \ (\{i, j\} \in E_n) \}.$$

The boolean quadric polytope, sometimes called the  $\mathcal{BQP}_n$ , arises naturally in quadratic 0-1 programming, and also has many applications in statistics, probability and theoretical physics (see Deza & Laurent [9]). Moreover, the  $\mathcal{BQP}_n$  of a graph  $G = (V_n, E)$  is a projection of a face of  $\text{BQP}_n$  (e.g., Padberg [21]).

Note that a pair  $(x, y)$  is an extreme point of  $\text{BQP}_n$  if and only if  $\mathcal{M}(x, y)$  is a symmetric rank one binary matrix. Therefore, the boolean quadric polytope is contained in the binary psd polytope. Moreover, the fact that binary psd matrices can be decomposed into the sum of symmetric rank one binary matrices can be used to show the following result:

**Proposition 7.**  $\text{BQP}_n \cap \{ a^T x + b^T y \leq 0 \} \cap \text{BQP}_n = \text{BQP}_n \cap \{ a^T x + b^T y \leq 0 \} \cap \mathcal{P}_n$

The homogeneous facets of a polyhedron are the facets that contain the origin, so that they are also facets of the cone that is generated by the incidence vectors of all feasible solutions. In the case of  $\text{BQP}_n$  and  $\mathcal{P}_n$ , this cone is sometimes called the  $\mathcal{BQP}_n$  (see again Deza & Laurent [9]).

In fact, the relationship between the boolean quadric and binary psd polytopes goes deeper than this.

**Proposition 8.**  $\text{BQP}_n \cap \mathcal{P}_{n+1}$

From Proposition 5, the following  $n + 1$  linear inequalities induce facets of  $\mathcal{P}_{n+1}$ :

$$\begin{aligned} x_{n+1} &\leq 1 \\ y_{i,n+1} &\leq x_i \ (i \in V_n). \end{aligned}$$

So consider the face of  $\mathcal{P}_{n+1}$  satisfying these inequalities at equality, and let  $(x^*, y^*) \in \{0, 1\}^{V_{n+1} \cup E_{n+1}}$  be a vertex of it. A consideration of the  $3 \times 3$  principal submatrices involving the last row/column shows that  $y_{ij}^* = x_i^* x_j^*$  for all  $\{i, j\} \in E_n$ . Thus, the matrix  $\mathcal{M}(x^*, y^*)$  is of the form

$$\begin{pmatrix} \tilde{x} \\ 1 \end{pmatrix} \begin{pmatrix} \tilde{x}^T & 1 \end{pmatrix} = \begin{pmatrix} \tilde{x} \tilde{x}^T & \tilde{x} \\ \tilde{x}^T & 1 \end{pmatrix},$$

where  $\tilde{x} \in \{0, 1\}^n$  is the vector obtained from  $x^*$  by dropping the last component. So,  $\mathcal{M}(x^*, y^*)$  is of rank one and there is a one-to-one correspondence between the symmetric rank one binary matrices and the vertices of the face. Thus,  $\text{BQP}_n$  is nothing but the projection of the face onto  $\mathbb{R}^{V_n \cup E_n}$ .  $\square$

The idea underlying the construction of the matrix  $\mathcal{M}(x^*, y^*)$  in the proof of Proposition 8 is due to Lovász & Schrijver [19] (see also Shor [22]), but the above polyhedral interpretation is new to our knowledge.

An immediate consequence of Proposition 8 is that valid or facet-inducing inequalities for  $\text{BQP}_n$  can be used to yield valid or facet-inducing inequalities for  $\mathcal{P}_{n+1}$ :

**Proposition 9.**

$$\sum_{i \in V_n} a_i x_i + \sum_{e \in E_n} b_e y_e \leq c$$

is valid for  $\text{BQP}_n$  if and only if  $\alpha \sum_{i \in V_n} (a_i + \beta_i) x_i - \alpha x_{n+1} + \sum_{e \in E_n} b_e y_e - \sum_{i \in V_n} \beta_i y_{i,n+1} \leq c - \alpha$  is valid for  $\mathcal{P}_{n+1}$ .

$$\sum_{i \in V_n} (a_i + \beta_i) x_i - \alpha x_{n+1} + \sum_{e \in E_n} b_e y_e - \sum_{i \in V_n} \beta_i y_{i,n+1} \leq c - \alpha,$$

$$\alpha \in \mathbb{R}, \beta \in \mathbb{R}^n$$

### 3.3 The Clique Partitioning Polytope

The clique partitioning polytope  $\text{PAR}_n$  (Grötschel & Wakabayashi [13]) of order  $n$  is defined as:

$$\text{PAR}_n = \text{conv} \{ y \in \{0, 1\}^{E_n} : y_{ik} + y_{jk} \leq y_{ij} + 1 \ (\forall \{i, j\} \in E_n, k \in V_n \setminus \{i, j\}) \}.$$

When a vector  $y \in \{0, 1\}^{E_n}$  belongs to  $\text{PAR}_n$ , it means that there exists a partition of  $V_n$  into sets such that, for all  $e \in E_n$ ,  $y_e = 1$  if and only if both end-vertices of  $e$  are in the same set. The clique partitioning polytope has applications in statistical clustering.

It is not hard to see that a vector  $y \in \{0, 1\}^{E_n}$  is a vertex of  $\text{PAR}_n$  if and only if there exists a vector  $x \in \{0, 1\}^{V_n}$  such that  $\mathcal{M}(x, y)$  is a binary psd matrix. Thus:

**Proposition 10.**  $\text{PAR}_n = \text{conv} \{ \mathcal{M}(x, y) : x \in \{0, 1\}^{V_n}, y \in \text{PAR}_n \} \subseteq \mathbb{R}^{E_n}$

In fact, we can say something stronger.

**Proposition 11.**  $\text{PAR}_n = \text{conv} \{ \mathcal{M}(x, y) : x \in \{0, 1\}^{V_n}, y \in \text{PAR}_n \} \subseteq \mathcal{P}_n$

Let  $\mathbf{1}_n$  denote the vector of  $n$  ones. Using Proposition 3, we have that a vector  $y \in \{0, 1\}^{E_n}$  is a vertex of  $\text{PAR}_n$  if and only if  $\mathcal{M}(\mathbf{1}_n, y)$  is a



binary psd matrix. Now, from Proposition 5, the following  $n$  linear inequalities induce facets of  $\mathcal{P}_n$ :

$$x_i \leq 1 \quad (i \in V_n).$$

Thus, the face of  $\mathcal{P}_n$  satisfying these  $n$  inequalities at equality, projected onto  $\mathbb{R}^{E_n}$ , is  $\text{PAR}_n$ . □

As in the previous subsection, this implies a lifting result:

**Proposition 12.** *Let  $b^T y \leq c$  be a linear inequality defining a facet of  $\text{PAR}_n$ . Then, the inequality  $\sum_{i \in V_n} \alpha_i x_i + b^T y \leq c + \sum_{i \in V_n} \alpha_i$  defines a facet of  $\mathcal{P}_n$  if and only if  $\alpha_i \leq 0$  for all  $i \in V_n$ .*

$$\sum_{i \in V_n} \alpha_i x_i + b^T y \leq c + \sum_{i \in V_n} \alpha_i,$$

$$\alpha_i \leq 0 \quad \forall i \in V_n$$

### 3.4 The Cut and Multicut Polytopes

Finally, we mention connections between the above polytopes and the  $J_n$  and  $\mathcal{P}_n$  polytopes.

Given any  $S \subseteq V_n$ , the set of edges

$$\delta_n(S) = \{\{i, j\} \in E_n : i \in S, j \in V_n \setminus S\}$$

is called an  $S$ -cut or simply cut. The  $\text{CUT}_n$  is the convex hull of the incidence vectors of all cuts in  $K_n$  (Barahona & Mahjoub [3]), i.e.,

$$\text{CUT}_n = \text{conv} \{y \in \{0, 1\}^{E_n} : \exists S \subset V_n : y_e = 1 \iff e \in \delta_n(S) (\forall \{i, j\} \in E_n)\}.$$

The cut polytope and the boolean quadric polytope are related via the so-called  $\text{CUT}_n$  lifting [9] which maps the boolean quadric polytope in  $\mathbb{R}^{E_n \cup V_n}$  to the cut polytope in  $\mathbb{R}^{E_{n+1}}$ . This means that there is a one-to-one correspondence between the facets of the respective polytopes. This correspondence is the following [9, Proposition 5.2.7]:

**Proposition 13.** *Let  $a \in \mathbb{R}^{V_n}$ ,  $b \in \mathbb{R}^{E_n}$ ,  $c \in \mathbb{R}^{E_{n+1}}$ . Then, the inequality  $a^T x + b^T y \leq c$  defines a facet of  $\text{CUT}_{n+1}$  if and only if  $a^T x + b^T y \leq a_0$  defines a facet of  $\text{BQP}_n$ .*

$$\begin{cases} c_{i,n+1} = a_i + \frac{1}{2} \sum_{j \in V_n \setminus \{i\}} b_{ij} & i \in V_n, \\ c_e = -\frac{1}{2} b_e & e \in E_n. \end{cases}$$

Let  $a_0 \in \mathbb{R}$ . Then, the inequality  $a^T x + b^T y \leq a_0$  defines a facet of  $\text{CUT}_{n+1}$  if and only if  $a^T x + b^T y \leq a_0$  defines a facet of  $\text{BQP}_n$ .

We remark that the cut polytope is also equivalent (under a simple linear mapping) to the convex hull of the psd matrices with  $\pm 1$  entries (see Goemans & Williamson [12], Laurent & Poljak [15]).

Now, given any partition of  $V_n$  into sets  $S_1, \dots, S_r$ , the set of edges

$$\delta_n(S_1, \dots, S_r) = \{\{i, j\} \in E_n : i \in S_p, j \in S_q \text{ for some } p \neq q\}$$

is called a *clique*. The *clique partitioning polytope*  $\text{MCUT}_n$  is defined accordingly (e.g., Deza [7]). It is not hard to see that the multicut polytope is nothing but the *projection* of the clique partitioning polytope:  $\text{MCUT}_n = \{\mathbf{1}_{E_n} - y \mid y \in \text{PAR}_n\}$ . As the incidence vectors in  $\text{MCUT}_n$  are just affine transformations of the incidence vectors in  $\text{PAR}_n$ , and vice versa, facets of one polytope are easily transformed to facets of the other. Suppose that the inequality  $b^T y \leq b_0$  defines a facet of  $\text{PAR}_n$  (respectively  $\text{MCUT}_n$ ). Substituting  $1 - y_e$  for  $y_e$  for all  $e \in E_n$  yields the inequality  $-b^T y \leq b_0 - \sum_{e \in E_n} b_e$ , which defines a facet of  $\text{MCUT}_n$  (respectively  $\text{PAR}_n$ ). We refer to the mapping  $y \mapsto \mathbf{1}_{E_n} - y$  between  $\text{PAR}_n$  and  $\text{MCUT}_n$  as *complement*.

Finally, we ‘complete the circle’ of results by establishing a link between the cut and clique partitioning polytopes:

**Proposition 14.** *The cut polytope  $\text{CUT}_n$  is the homogenization of the multicut polytope  $\text{MCUT}_n$ .*

This fact was pointed out in Deza [7].

In Fig. 1 we summarize the relationships between the five polytopes as established by Propositions 8 to 14. (Note that Proposition 7 is not displayed.) As we remarked in the introduction, the binary psd polytope is the most complex of the five polytopes under discussion. We point out however that the multicut and clique partitioning polytopes are themselves more complex than the cut and

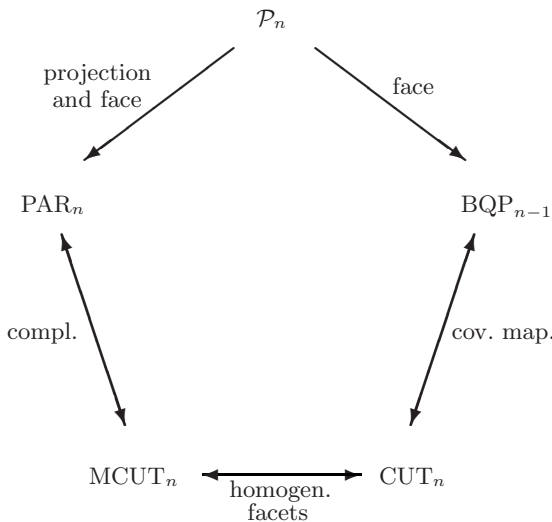


Fig. 1. A pentagon of polyhedral relations

boolean quadric polytopes: the latter polytopes exhibit a high degree of symmetry, via the so-called  $\vee, \wedge, \vee, \wedge$  operation, which enables one to derive all facets of  $\text{CUT}_n$  or  $\text{BQP}_n$  given a list of only the homogeneous facets (see Barahona & Mahjoub [3] and Deza & Laurent [9]). Thus, a complete description of  $\text{CUT}_n$  and  $\text{BQP}_{n-1}$  can be obtained from a complete description of  $\text{MCUT}_n$  or  $\text{PAR}_n$ , which in turn can be obtained from a complete description of the binary psd polytope  $\mathcal{P}_n$ .

## 4 Valid Inequalities

We now move on to consider some specific classes of valid inequalities for the binary psd polytope, and show how they imply existing known results for the other polytopes mentioned above.

### 4.1 Some Simple Results

In Proposition 5 we pointed out that the upper bounds  $x_i \leq 1$ , the non-negativity inequalities  $y_e \geq 0$  for all  $e \in E_n$ , and the inequalities (3) and (4) induce facets of the binary psd polytope  $\mathcal{P}_n$ . All of these inequalities were shown by Padberg to induce facets of the boolean quadric polytope  $\text{BQP}_n$ . Moreover, the inequalities (4) imply, via Proposition 8, the validity of the following inequalities for  $\text{BQP}_n$ :

$$x_i + x_j \leq 1 + y_{ij} \quad (\{i, j\} \in E_n).$$

These inequalities too were proved to induce facets of  $\text{BQP}_n$  by Padberg.

In a similar way, Propositions 5 and 11 show that the inequalities  $y_e \geq 0$  for all  $e \in E_n$ , and the inequalities

$$y_{ik} + y_{jk} \leq 1 + y_{ij} \quad (\{i, j\} \in E_n, k \in V_n \setminus \{i, j\}),$$

are valid for the clique partitioning polytope  $\text{PAR}_n$ . These inequalities were proved to induce facets of  $\text{PAR}_n$  by Grötschel and Wakabayashi [13].

### 4.2 Hypermetric Correlation Inequalities

If we apply the first definition of psd-ness given in the introduction to the matrix  $\mathcal{M}(x, y)$ , we see that the following inequalities are valid for  $\mathcal{P}_n$ :

$$\sum_{i \in V_n} a_i^2 x_i + 2 \sum_{\{i, j\} \in E_n} a_i a_j y_{ij} \geq 0 \quad (\forall a \in \mathbb{R}^n). \tag{5}$$

It follows from known results on the cut and correlation cones, however, that these inequalities do not define facets of  $\mathcal{P}_n$ . Indeed, the following  $\vee, \wedge, \vee, \wedge$  inequalities are well-known to be valid for the cut cone (see Deza & Laurent [9]):

$$\sum_{\{i, j\} \in E_n} a_i a_j y_{ij} \leq 0 \quad (\forall a \in \mathbb{Z}^n : \sum_{i=1}^n a_i = 1). \tag{6}$$

Under the covariance mapping, they correspond to the following inequalities, which are valid for the correlation cone:

$$\sum_{i \in V_n} a_i(a_i - 1)x_i + 2 \sum_{\{i,j\} \in E_n} a_i a_j y_{ij} \geq 0 \quad (\forall a \in \mathbb{Z}^n). \tag{7}$$

We will follow Deza & Grishukhin [5] in calling them hypermetric correlation inequalities. Note that the hypermetric correlation inequalities, being homogeneous, are valid for  $\text{BQP}_n$ . Then, by Proposition 7, they are valid for  $\mathcal{P}_n$  as well. They are easily shown to dominate the inequalities (5).

The hypermetric and hypermetric correlation inequalities have been studied in depth by Deza and colleagues (e.g., [4,5,6,8,9]). Conditions under which the hypermetric inequalities induce facets of the cut cone are surveyed in [9]. Via the covariance mapping, one can derive analogous conditions under which the hypermetric correlation inequalities induce facets of the correlation cone, and therefore of  $\text{BQP}_n$ . By Proposition 7, they induce facets of  $\mathcal{P}_n$  under the same conditions.

Note that the non-negativity inequalities  $y_e \geq 0$  for all  $e \in E_n$ , and the inequalities (3) and (4), are hypermetric correlation inequalities.

An important result, which will be of relevance in what follows, is that the hypermetric inequalities define a polyhedral cone [6]. That is, although the inequalities (6) are infinite in number, there exists a finite subset of them that dominates all the others. Via the covariance mapping, the hypermetric correlation inequalities also define a polyhedral cone.

Now, we consider the implications of moving ‘clockwise’ in Fig. 1. The hypermetric correlation inequalities (7) imply, via Proposition 8, the validity of the following inequalities for  $\text{BQP}_n$ :

$$\sum_{i \in V_n} a_i(2b + a_i - 1)x_i + 2 \sum_{\{i,j\} \in E_n} a_i a_j y_{ij} \geq b(1 - b) \quad (\forall a \in \mathbb{Z}^n, b \in \mathbb{Z}). \tag{8}$$

These inequalities, which include the hypermetric correlation inequalities as a special case, are also well-known in the literature [9]. Most of the inequalities shown by Padberg [21] to induce facets of  $\text{BQP}_n$  — such as the  $k$ -set inequalities — are in fact special cases of the inequalities (8).

Under the covariance mapping, the inequalities (8) correspond to the following inequalities for  $\text{CUT}_n$ :

$$\sum_{\{i,j\} \in E_n} a_i a_j y_{ij} \leq \lfloor \sigma(a)^2 / 4 \rfloor \quad (\forall a \in \mathbb{Z}^n : \sigma(a) \text{ odd}), \tag{9}$$

where  $\sigma(a) = \sum_{i \in V_n} a_i$ . These inequalities, which include the hypermetric inequalities as a special case, are sometimes called  $k$ -set inequalities (see Deza & Laurent [9], Avis & Umemoto [2]). They induce facets of  $\text{CUT}_n$  if and only if they can be obtained from facet-inducing hypermetric inequalities via the switching operation.

We now present several new results. (Detailed proofs will be given in the full version of the paper.) The first two results show that the separation problems

for the inequalities (8) and (9) can be reduced to the separation problems for the inequalities (7) and (6), respectively. They can be proved either directly or by using Proposition 8 and the covariance mapping.

**Proposition 15.**  $(x^*, y^*) \in [0, 1]^{V_n \cup E_n}$ ,  $(x', y') \in [0, 1]^{V_{n+1} \cup E_{n+1}}$ ,  $x'_i = x^*_i$ ,  $i \in V_n$ ,  $x'_{n+1} = 1$ ,  $y'_e = y^*_e$ ,  $e \in E_n$ ,  $y'_{i,n+1} = x^*_i$ ,  $i \in V_n$ ,  $(x^*, y^*) \in \mathbb{R}^{V_n \cup E_n}$ ,  $(x', y') \in \mathbb{R}^{V_{n+1} \cup E_{n+1}}$ .  $\square$

**Proposition 16.**  $y^* \in [0, 1]^{E_n}$ ,  $y' \in [0, 1]^{E_{n+1}}$ ,  $y'_e = y^*_e$ ,  $e \in E_n$ ,  $y'_{i,n+1} = 1 - y^*_{i,n}$ ,  $i \in V_{n-1}$ ,  $y'_{n,n+1} = 1$ ,  $y^* \in \mathbb{R}^{E_n}$ ,  $y' \in \mathbb{R}^{E_{n+1}}$ .  $\square$

Unfortunately, the complexity of separation for the hypermetric inequalities is unknown (see Avis [1]). Nevertheless, the above two results have interesting polyhedral implications:

**Proposition 17.**  $\mathbb{R}^{V_{n+1} \cup E_{n+1}}$ ,  $y_{i,n+1} = x_i$ ,  $i \in V_n$ ,  $x_{n+1} = 1$ ,  $\mathbb{R}^{V_n \cup E_n}$ .  $\square$

**Proposition 18.**  $\mathbb{R}^{E_{n+1}}$ ,  $y_{i,n} + y_{i,n+1} = 1$ ,  $i \in V_{n-1}$ ,  $y_{n,n+1} = 1$ ,  $\mathbb{R}^{E_n}$ .  $\square$

Proposition 18 implies the following result, which answers in the affirmative a question raised by Avis & Umemoto [2]:

**Proposition 19.**  $k \in \mathbb{Z}^n$ .  $\square$

The hypermetric cone is polyhedral. The intersection of a polyhedral cone with an affine subspace is also polyhedral, and so is its projection onto any subspace. The result then follows from Proposition 18.  $\square$

For similar reasons, the inequalities (8) also define a polytope.

To close this subsection, we consider moving ‘anticlockwise’ in Fig. 1. The inequalities (7) imply, via Proposition 11, the validity of the following inequalities for  $\text{PAR}_n$ :

$$\sum_{\{i,j\} \in E_n} a_i a_j y_{ij} \geq \frac{1}{2} \sum_{i \in V_n} a_i (1 - a_i) \quad (\forall a \in \mathbb{Z}^n), \tag{10}$$

and the following inequalities for  $\text{MCUT}_n$ :

$$\sum_{\{i,j\} \in E_n} a_i a_j y_{ij} \leq \sigma(a)(\sigma(a) - 1)/2 \quad (\forall a \in \mathbb{Z}^n).$$

The validity of these inequalities for  $\text{MCUT}_n$ , and the fact that they induce facets under certain conditions, was also observed by Deza & Laurent [9] (p. 465). We remark that, when  $a_i$  is binary for all vertices apart from one, the inequalities (10) reduce to the so-called  $(s, T)$  inequalities of Oosten [20], shown to induce facets under certain conditions.

### 4.3 Gap Inequalities

We now present a class of valid inequalities that dominate the hypermetric correlation inequalities (7).

**Proposition 20.** *For all  $a \in \mathbb{Z}^n$ , the following inequality is valid for  $\mathcal{P}_n$ .*

$$\sum_{i \in V_n} a_i(a_i - a_{\min})x_i + 2 \sum_{\{i,j\} \in E_n} a_i a_j y_{ij} \geq 0 \quad (\forall a \in \mathbb{Z}^n), \quad (11)$$

$$a_{\min} = \min \{a^T x : a^T x > 0, x \in \{0, 1\}^n\}.$$

From the definition of  $a_{\min}$ , we have that, for any  $x \in \{0, 1\}^n$ , either  $a^T x \leq 0$  or  $a^T x \geq a_{\min}$ . In either case, we have  $a^T x(a^T x - a_{\min}) \geq 0$ . Thus, when  $\mathcal{M}(x, y)$  is a symmetric rank one binary matrix, we have  $a^T \mathcal{M}(x, y)a - a_{\min} a^T x \geq 0$ . This establishes the validity of the inequalities (11) for  $\text{BQP}_n$ . By Proposition 7, they are also valid for  $\mathcal{P}_n$ .  $\square$

These strengthened inequalities can be shown to imply (via Proposition 8 and the covariance mapping) the validity of the so-called  $(s, T)$  inequalities for  $\text{CUT}_n$  (Laurent & Poljak [16]). They also imply some new inequalities for  $\text{BQP}_n$ ,  $\text{PAR}_n$  and  $\text{MCUT}_n$ . Details will be given in the full version of the paper.

### 4.4 Inequalities Related to Cycles and Paths

We have discovered several classes of facet-inducing inequalities of  $\mathcal{P}_n$  that are related to cycles and paths in  $K_n$ . We mention two such classes here. The first class we consider is of interest because the inequalities are inhomogeneous and involve only the  $y$  variables. Let  $C \subset E_n$  be the edge set of a simple cycle of odd length at least 5, and let  $\bar{C}$  be the set of 2-chords of  $C$ . The

$$\sum_{e \in C} y_e - \sum_{e \in \bar{C}} y_e \leq (|C| - 1)/2 \quad (12)$$

is shown in Grötschel & Wakabayashi [13] to be facet-inducing for the clique partitioning polytope  $\text{PAR}_n$ . This inequality can be lifted, according to Proposition 12, so that we can establish the following.

**Proposition 21.** *For all  $a \in \mathbb{Z}^n$ , the following inequality is valid for  $\mathcal{P}_n$ .*  $\square$

The existence of such inhomogeneous facet-inducing inequalities for  $\mathcal{P}_n$  implies that  $\mathcal{P}_n$  is strictly contained in the intersection of the correlation cone and the unit hypercube. We have found some other interesting inhomogeneous facet-inducing inequalities for  $\mathcal{P}_n$ , which will be mentioned in the full version of the paper.

Next, however, we consider the class of 2-chorded path inequalities. Let  $P = \{e = \{i, i + 1\} : i = 1, \dots, k - 1\}$  be the edge set of a path of length  $k - 1 \geq 2$  and let  $\bar{P} = \{e = \{i, i + 2\} : i = 1, \dots, k - 2\}$  be the set of 2-chords of  $P$ . Denote by  $I^- = \{i \in V_n(P) : (i \bmod 2) = 1\}$  and  $I^+ = \{i \in V_n(P) : (i \bmod 2) = 0\}$  the odd and even endnodes of the edges in  $P$ , respectively. Let  $Z \subseteq V_n \setminus V_n(P)$  be nonempty and define  $R = \delta_n(I^+, Z)$  and  $\bar{R} = \delta_n(I^-, Z)$ . The following 2-chorded path inequality

$$\sum_{e \in P \cup R} y_e - \sum_{e \in \bar{P} \cup \bar{R}} y_e - \left\lfloor \frac{k+2}{4} \right\rfloor \sum_{\{i,j\} \subseteq Z} y_{ij} \leq |I^+|$$

is shown in [23] to define a facet of  $\text{PAR}_n$  under mild conditions. The fact that we allow for  $|Z| \geq 1$  provides a generalization of the 2-chorded path inequality that was originally introduced in [13]. In the original inequality it is assumed that  $|Z| = 1$ , and that inequality only induces a facet of  $\text{PAR}_n$  when the path has even length  $k - 1$ . We note that a similar generalization of the inequality is considered in [20], where the variables  $y_{ij}$  with  $\{i, j\} \subseteq Z$  have  $-1$  coefficients. That inequality is mistakenly claimed to be valid for  $\text{PAR}_n$ .

When the above inequality is lifted according to Proposition [12] we obtain

$$- \sum_{i \in I^+} x_i + \sum_{e \in P \cup R} y_e - \sum_{e \in \bar{P} \cup \bar{R}} y_e - \left\lfloor \frac{k+2}{4} \right\rfloor \sum_{\{i,j\} \subseteq Z} y_{ij} \leq 0. \tag{13}$$

We have the following result.

**Proposition 22.** □  $\forall k \geq 2, \forall Z \subseteq V_n \setminus V_n(P), \forall \alpha \in \mathbb{R}_+^n, |Z| \geq 2, \forall \alpha \in \mathbb{R}_+^n, (|P| \bmod 2) = 0$

Note that these inequalities are homogeneous, and therefore induce facets of the correlation cone as well.

### 4.5 Lifting Facets of the Clique Partitioning Polytope

Finally, we would like to mention that sometimes we are able to obtain new facets of the clique partitioning polytope  $\text{PAR}_{n+1}$  of order  $n + 1$  from known facets of  $\text{PAR}_n$  via the other polytopes considered here. This involves the following steps.

Suppose that  $b^T y \leq b_0$  is an inequality that induces a facet of  $\text{PAR}_n$ . This inequality can be lifted (Proposition [12]) to a facet-inducing inequality  $-\alpha^T x + b^T y \leq b_0 - \alpha^T \mathbf{1}_n$  for  $\mathcal{P}_n$ , where  $\alpha \in \mathbb{R}_+^n$ . Whenever this lifted inequality for the binary psd polytope  $\mathcal{P}_n$  is homogeneous, i.e.,  $\alpha^T \mathbf{1}_n = b_0$ , it also induces

a facet of the boolean quadric polytope  $BQP_n$  (Proposition 7). The inequality  $c^T y \leq 0$  obtained via the covariance mapping (Proposition 13) defines a facet of the cut and multicut polytopes  $CUT_{n+1}$  and  $MCUT_{n+1}$  (Proposition 14). Then, by complementing this latter inequality, we obtain a facet-inducing inequality  $c^T y \geq \sum_{e \in E_{n+1}} c_e$  for  $PAR_{n+1}$ .

Examples of new facets of  $PAR_{n+1}$  obtained in this manner will be given in the full version of the paper.

## References

1. Avis, D.: On the complexity of testing hypermetric, negative type,  $k$ -gonal and gap inequalities. In: Akiyama, J., Kano, M. (eds.) JCDCG 2002. LNCS, vol. 2866. Springer, Heidelberg (2003)
2. Avis, D., Umemoto, J.: Stronger linear programming relaxations for max-cut. *Math. Program.* 97, 451–469 (2003)
3. Barahona, F., Mahjoub, A.R.: On the cut polytope. *Math. Program.* 36, 151–173 (1986)
4. Deza, M., Dutour, M.: The hypermetric cone on seven vertices. *Experimental Math.* 12, 433–440 (2003)
5. Deza, M., Grishukhin, V.P.: Voronoi L-decomposition of  $PSD_n$  and the hypermetric correlation cone. In: Engel, P., Syta, H. (eds.) *Voronoi's Impact on Modern Science*, Kiev, Institute of Mathematics of the National Academy of Science, Ukraine (1998)
6. Deza, M., Grishukhin, V.P., Laurent, M.: The hypermetric cone is polyhedral. *Combinatorica* 13, 397–411 (1993)
7. Deza, M., Grötschel, M., Laurent, M.: Clique-web facets for multicut polytopes. *Math. Oper. Res.* 17, 981–1000 (1992)
8. Deza, M.M., Laurent, M.: Facets for the cut cone I. *Math. Program.* 56, 121–160 (1992)
9. Deza, M.M., Laurent, M.: *Geometry of Cuts and Metrics*. Springer, Berlin (1997)
10. Dukanovic, I., Rendl, F.: Semidefinite programming relaxations for graph coloring and maximal clique problems. *Math. Program.* 109, 345–365 (2007)
11. Goemans, M.X.: Semidefinite programming in combinatorial optimization. *Math. Program.* 79, 143–161 (1997)
12. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. of the ACM* 42, 1115–1145 (1995)
13. Grötschel, M., Wakabayashi, Y.: Facets of the clique partitioning polytope. *Math. Program.* 47, 367–387 (1990)
14. Hill, R.D., Waters, S.R.: On the cone of positive semidefinite matrices. *Lin. Alg. Appl.* 90, 81–88 (1987)
15. Laurent, M., Poljak, S.: On a positive semidefinite relaxation of the cut polytope. *Lin. Alg. Appl.* 223/224, 439–461 (1995)
16. Laurent, M., Poljak, S.: Gap inequalities for the cut polytope. *SIAM J. Matrix Anal.* 17, 530–547 (1996)
17. Laurent, M., Rendl, F.: Semidefinite programming and integer programming. In: Aardal, K., Nemhauser, G., Weismantel, R. (eds.) *Handbook on Discrete Optimization*, pp. 393–514. Elsevier, Amsterdam (2005)



18. Lovász, L.: On the Shannon capacity of a graph. *IEEE Trans. Inf. Th.* 25, 1–7 (1979)
19. Lovász, L., Schrijver, A.J.: Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Opt.* 1, 166–190 (1991)
20. Oosten, M., Rutten, J.H.G.C., Spieksma, F.C.R.: The clique partitioning problem: facets and patching facets. *Networks* 38, 209–226 (2001)
21. Padberg, M.W.: The boolean quadric polytope: some characteristics, facets and relatives. *Math. Program.* 45, 139–172 (1989)
22. Shor, N.Z.: Quadratic optimization problems. *Tekhnicheskaya Kibernetika* 1, 128–139 (1987)
23. Sørensen, M.M.: A polyhedral approach to graph partitioning. PhD thesis, Aarhus School of Business (1995), available at <http://www.hha.dk/~mim/>

# Vertex Cover Resists SDPs Tightened by Local Hypermetric Inequalities\*

Konstantinos Georgiou, Avner Magen, and Iannis Tourlakis

Department of Computer Science, University of Toronto  
{cgeorg, avner, iannis}@cs.toronto.edu

**Abstract.** We consider the standard semidefinite programming (SDP) relaxation for VERTEX COVER to which all hypermetric inequalities supported on at most  $k$  vertices have been added. We show that the integrality gap for such SDPs remains  $2 - o(1)$  as long as  $k = O(\sqrt{\log n / \log \log n})$ . This extends successive results by Kleinberg-Goemans, Charikar and Hatami et al. which analyzed integrality gaps of the standard VERTEX COVER SDP relaxation as well as for SDPs tightened using triangle and pentagonal inequalities.

Our result is complementary but incomparable to a recent result by Georgiou et al. proving integrality gaps for VERTEX COVER SDPs in the Lovász-Schrijver hierarchy. One of our contributions is making explicit the difference between the SDPs considered by Georgiou et al. and those analyzed in the current paper. We do this by showing that VERTEX COVER SDPs in the Lovász-Schrijver hierarchy fail to satisfy any hypermetric constraints supported on independent sets of the input graph.

## 1 Introduction

A *vertex cover* for a graph is a subset of vertices that touches all edges in the graph. Determining the approximability of the minimum VERTEX COVER problem on graphs is one of the outstanding problems in theoretical computer science. While there exists a trivial 2-approximation algorithm, considerable efforts have failed to obtain an approximation ratio better than  $2 - o(1)$ . On the other hand, the strongest PCP-based hardness result known [8] only shows that 1.36-approximation of VERTEX COVER is NP-hard. Only by assuming Khot's Unique Game Conjecture [15] can it be shown that  $2 - o(1)$ -approximation is NP-hard [16].

Several recent papers [17, 12, 4, 14, 13, 10] examine whether semidefinite programming (SDP) relaxations of VERTEX COVER might yield better approximations. Goemans and Williamson [11] introduced semidefinite programming relaxations as an algorithmic technique using it to obtain a 0.878-approximation for MAX-CUT. Since then semidefinite programming has arguably become our most powerful tool for designing approximation algorithms. Indeed, for many NP-hard optimization problems the best approximation ratios are achieved using SDP-based algorithms.

---

\* Funded in part by NSERC.

Given a graph  $G = (V, E)$ , the standard SDP relaxation for VERTEX COVER is

$$\begin{aligned} \min \quad & \sum_{i \in V} (1 + \mathbf{v}_0 \cdot \mathbf{v}_i) / 2 \\ \text{s.t.} \quad & (\mathbf{v}_0 - \mathbf{v}_i) \cdot (\mathbf{v}_0 - \mathbf{v}_j) = 0 \quad \forall i, j \in E \\ & \|\mathbf{v}_i\| = 1 \quad \forall i \in \{0\} \cup V \end{aligned} \tag{1}$$

Halperin [12] employed this relaxation together with an appropriate rounding technique to obtain a  $(2 - \Omega(\log \log \Delta / \log \Delta))$ -approximation for VERTEX COVER for graphs with maximal degree  $\Delta$ . Unfortunately, Kleinberg and Goemans [17] showed that in general this relaxation has an integrality gap of  $2 - o(1)$ .

One possible avenue for decreasing this integrality gap comes from the following simple observation: for any integral (or rather, one-dimensional) solution,  $\|\mathbf{v}_i - \mathbf{v}_j\|^2$  is an  $\ell_1$  metric. Therefore the addition of inequalities on the distances  $\|\mathbf{v}_i - \mathbf{v}_j\|^2$  that are valid for  $\ell_1$  metrics may yield a possible tightening of the SDP (note that the constraint  $(\mathbf{v}_0 - \mathbf{v}_i) \cdot (\mathbf{v}_0 - \mathbf{v}_j) = 0$  in SDP (1) is in fact the following distance constraint “in disguise”:  $\|\mathbf{v}_i - \mathbf{v}_0\|^2 + \|\mathbf{v}_j - \mathbf{v}_0\|^2 = \|\mathbf{v}_i - \mathbf{v}_j\|^2$ ).

For example, since  $\ell_1$  metrics satisfy the triangle inequality, we could add the following constraint to SDP (1):

$$\|\mathbf{v}_i - \mathbf{v}_j\|^2 + \|\mathbf{v}_j - \mathbf{v}_k\|^2 \geq \|\mathbf{v}_i - \mathbf{v}_k\|^2 \quad \forall i, j, k \in \{0\} \cup V. \tag{2}$$

This  $\ell_2^2$  triangle inequality plays a crucial role in the breakthrough Arora-Rao-Vazirani SPARSEST CUT algorithm [2]. This suggests that the addition of inequalities satisfied by  $\ell_1$  metrics may be exactly what is needed to get a  $2 - \Omega(1)$  approximation for VERTEX COVER.

Indeed, Hatami et al. [13] prove that if SDP (1) is strengthened by requiring that the distances  $\|\mathbf{v}_i - \mathbf{v}_j\|^2$  satisfy  $\dots$   $\ell_1$  inequalities (i.e., the vectors  $\mathbf{v}_i$  equipped with the  $\ell_2^2$  norm  $\|\cdot\|^2$  are  $\ell_1$ -embeddable), then the resulting relaxation has no integrality gap. Of course, the caveat here is that the resulting relaxation has exponentially many constraints and is hence intractable. To obtain a tractable relaxation (or at least one computable in subexponential time), our relaxation must use only a limited subset of  $\ell_1$  inequalities.

One canonical subclass of  $\ell_1$  inequalities is the discrete and easily-described class of  $k$ -gonal inequalities (see the Preliminaries for definitions). These include the triangle inequalities as well as the so-called pentagonal, heptagonal, etc., inequalities. That such inequalities might be useful for designing improved approximation algorithms is illustrated, for example, in a work by Avis and Umemoto [3]. Avis and Umemoto show that for dense graphs, linear programming relaxations of MAX CUT based on the  $k$ -gonal inequalities have integrality gap at most  $1 + 1/k$ . This, in a sense, gives rise to an LP-based PTAS for MAX CUT.

Unfortunately, for VERTEX COVER Charikar [4] showed that even with the addition of the triangle inequality (2) the integrality gap of SDP (1) remains  $2 - o(1)$ . However, Karakostas [14] did show that adding the triangle inequality (as well as the “antipodal” triangle inequalities  $(\pm \mathbf{v}_i - \pm \mathbf{v}_j) \cdot (\pm \mathbf{v}_i - \pm \mathbf{v}_k) \geq 0$ ) yields a  $(2 - \Omega(1/\sqrt{\log n}))$ -approximation for VERTEX COVER, currently the best ratio achievable by any algorithm. Hatami et al. [13] subsequently showed that Karakostas’s SDP even with the addition of the pentagonal inequalities has integrality gap  $2 - o(\sqrt{\log \log n / \log n})$ .

In this work we rule out the possibility that adding local hypermetric constraints improves the integrality gap of VERTEX COVER SDPs:

**Theorem 1.** *The integrality gap of VERTEX COVER SDPs with  $O(\sqrt{\log n / \log \log n})$  local hypermetric constraints is  $2 - o(1)$ .*

As mentioned above, Hatami et al. [13] show that adding the constraint that solutions to SDP (1) be  $\ell_1$ -embeddable results in an SDP with no integrality gap. Theorem 1 then immediately gives the following corollary about  $\ell_2^2$  metrics:

**Corollary 1.** *There exists a metric space with  $\ell_2^2$  local hypermetric constraints such that its integrality gap is  $O(\sqrt{\log n / \log \log n})$  and it does not embed isometrically into  $\ell_1$ .*

It is interesting to compare Corollary 1 with recent results contrasting local and global phenomena in metric spaces. In [15] the authors describe metric spaces that cannot be well-embedded into  $\ell_1$  but locally every small subset embeds isometrically into  $\ell_1$ . In contrast, our corollary shows the existence of a metric that locally resembles  $\ell_1$  (although not provably  $\ell_1$ ) but globally does not embed isometrically into  $\ell_1$ . From this standpoint, this is far weaker than [15]. However, the metric we supply is also an  $\ell_2^2$  metric. Finding  $\ell_2^2$  metrics that are far from being  $\ell_1$  proved to be a very challenging task (see Khot and Vishnoi’s work [6] motivated by integrality gap instances for SPARSEST CUT). To the best of our knowledge, there are no known results that point to such metrics which further satisfy any local conditions beyond the obvious triangle inequality.

A result related to Theorem 1 was proved by Georgiou et al. in [10]. The main result of that paper showed that SDP relaxations obtained by tightening the standard linear programming relaxation for VERTEX COVER using  $O(\sqrt{\log n / \log \log n})$  rounds of the  $LS_+$  “lift-and-project” method of Lovász and Schrijver [18] have integrality gap  $2 - o(1)$ . The SDPs considered in [10] seem intimately related to those obtained by adding local  $\ell_1$  or hypermetric constraints. Indeed, it is well known that relaxations from the LP Lovász-Schrijver hierarchy satisfy all valid local LP constraints. However, it is also known [10] that relaxations from the SDP Lovász-Schrijver hierarchy do not necessarily satisfy all valid local SDP constraints. In particular, the VERTEX COVER SDP relaxation obtained after  $k$  rounds of the  $LS_+$  method is not obviously comparable to the relaxation obtained by adding all order- $k$  hypermetric inequalities to SDP (1). In section 4 we show in a strong sense the incomparability of these relaxations: Fix any subset  $S$  of vertices that is an independent set in the underlying graph. We then find a hypermetric inequality supported on all points of  $S$  that is nevertheless not valid for any VERTEX COVER SDP in the Lovász-Schrijver hierarchy. In particular, this shows that the integrality gaps proved in [10] do not preclude the possibility that adding such concrete constraints as, say, the “heptagonal” inequalities, may result in an improved SDP relaxation.

We briefly describe how we prove Theorem 1. We use the same graph family as in [17, 4, 13, 10]. The SDP solution can be thought of as an  $\ell_1$  metric to which a small perturbation is applied. This perturbation is characterized by two “infinitesimal” parameters,  $\gamma$  and  $\epsilon$ , relating to the graph and the integrality

gap, respectively. We show that hypermetric inequalities that are supported on  $k \geq 4$  points, one of which is  $\mathbf{v}_0$ , must have a slack component that depends on  $k$  and on  $\epsilon$  and  $\gamma$ , that will be maintained as long as  $k\gamma = O(\epsilon)$ . The case when  $k = 3$  (i.e., the triangle inequality) is covered by [4] and [13], and the case where  $\mathbf{v}_0$  does not participate in the inequality is handled by the fact that the metric formed by the remaining vectors is an  $\ell_1$  metric. Setting  $\epsilon$  to an arbitrary small constant, and setting  $\gamma$  to  $\Theta(\sqrt{\log \log n / \log n})$  provides the bound in our theorem.

## 2 Preliminaries

Given two vectors  $\mathbf{x}, \mathbf{y} \in \{-1, 1\}^n$  their Hamming distance  $d_H(\mathbf{x}, \mathbf{y})$  is  $|\{i \in [n] : x_i \neq y_i\}|$ . For two vectors  $\mathbf{u} \in \mathbb{R}^n$  and  $\mathbf{v} \in \mathbb{R}^m$  denote by  $(\mathbf{u}, \mathbf{v}) \in \mathbb{R}^{n+m}$  the vector whose projection on the first  $n$  coordinates is  $\mathbf{u}$  and on the last  $m$  coordinates is  $\mathbf{v}$ .

The tensor product  $\mathbf{u} \otimes \mathbf{v}$  of vectors  $\mathbf{u} \in \mathbb{R}^n$  and  $\mathbf{v} \in \mathbb{R}^m$  is the vector in  $\mathbb{R}^{nm}$  indexed by ordered pairs from  $n \times m$  and that assumes the value  $u_i v_j$  at coordinate  $(i, j)$ . Define  $\mathbf{u}^{\otimes d}$  to be the vector in  $\mathbb{R}^{n^d}$  obtained by tensoring  $\mathbf{u}$  with itself  $d$  times. Let  $P(x) = c_1 x^{t_1} + \dots + c_q x^{t_q}$  be a polynomial with nonnegative coefficients. Then  $T_P$  is the function that maps a vector  $\mathbf{u}$  to the vector  $T_P(\mathbf{u}) = (\sqrt{c_1} u^{\otimes t_1}, \dots, \sqrt{c_q} u^{\otimes t_q})$ .

**Fact:** For all vectors  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ ,  $T_P(\mathbf{u}) \cdot T_P(\mathbf{v}) = P(\mathbf{u} \cdot \mathbf{v})$ .

We quickly review the facts we need about  $\ell_1$  inequalities. Deza and Laurent [7] is a good source for more information.

A finite metric space is an  $\ell_1$  metric space if it can be embedded in  $\ell_1$ -normed space so that all distances remain unchanged. It is easy to see that the set  $\mathcal{C}$  of all  $\ell_1$  metrics on a fixed number of points is a convex cone. Let  $X$  be a set of size  $n$ . A subset  $S$  of  $X$  is associated with a metric  $\delta_S(x, y)$  that is called a cut metric, and is defined as  $|\chi_S(x) - \chi_S(y)|$ , where  $\chi_S(\cdot)$  is the characteristic function of  $S$ . These metrics are the extreme rays of  $\mathcal{C}$ ; namely, every  $\ell_1$  metric is a positive linear combination of cut metrics. This fact leads to a simple characterization of all inequalities that are valid for  $\ell_1$  metrics as follows. Consider the polar cone of  $\mathcal{C}$ ,

$$\mathcal{C}^* = \{B \in \mathbb{R}^{n \times n} \mid B \cdot D \leq 0 \text{ for all } D \in \mathcal{C}\},$$

where by  $B \cdot D$  we denote the matrix inner product of  $B$  and  $D$ , that is  $B \cdot D = \text{trace}(BD^t) = \sum_{i,j} B_{ij} D_{ij}$ . Notice that for  $B$  to be in  $\mathcal{C}^*$  it is enough to require that  $B \cdot \delta_S \leq 0$  for all cuts  $S$ . By definition it is clear that any  $B \in \mathcal{C}^*$  defines a valid inequality such that  $\sum_{i,j} B_{ij} d_{ij} \leq 0$  whenever  $d$  is an  $\ell_1$  metric. Conversely, (strong) duality implies that if  $d$  satisfies all inequalities of this type for every  $B \in \mathcal{C}^*$  then  $d$  is an  $\ell_1$  metric.

A special canonical class of  $\ell_1$  inequalities is the class of  $\mathbf{b}$ -inequalities. Let  $\mathbf{b} \in \mathbb{Z}^k$ , with  $\sum_{i=1}^k b_i = 1$ . It can be easily verified that  $B = \mathbf{b}\mathbf{b}^t$  is in  $\mathcal{C}^*$ . The inequality  $\sum_{i,j} b_i b_j d_{ij} \leq 0$  is called a  $\mathbf{b}$ -inequality. If we further

require  $\mathbf{b} \in \{-1, 1\}^k$ , in which case the hypermetric is called  $\bullet \dots$ , we obtain the  $k$ -gonal inequalities, e.g., the triangle inequality for  $k = 3$ , pentagonal inequality for  $k = 5$ , etc.

### 3 Construction and Proof

Fix arbitrarily small constants  $\gamma, \epsilon > 0$  such that  $\epsilon > 3\gamma$ , and let  $m$  be a sufficiently large integer. The  $G_m^\gamma$  is the graph with vertices  $\{-1, 1\}^m$  and where two vertices  $i, j \in \{-1, 1\}^m$  are adjacent if  $d_H(i, j) = (1 - \gamma)m$ . A classical result of Frankl and Rödl [9] implies that the size of a minimal vertex cover in  $G_m^\gamma$  is  $2^m(1 - o(1))$  whenever  $\gamma = \Omega(\sqrt{\log m/m})$ . We denote the vertices  $V$  of  $G$  as vectors  $\mathbf{w}_i \in \{-1, 1\}^m$  (the association of index  $i$  with a vector in the cube is arbitrary) and normalize these to get unit vectors  $\mathbf{u}_i = \frac{1}{\sqrt{m}}\mathbf{w}_i$ .

Consider the polynomial

$$P(x) = \beta x(x + 1)^{\frac{2m}{\gamma}} + \alpha x^{\frac{1}{\gamma}} + (1 - \alpha - 2\beta)x,$$

where the constants  $\alpha, \beta > 0$  will be defined below. Let  $\mathbf{z}_0 = (1, 0 \dots, 0)$ ,  $\mathbf{z}_i = (2\epsilon, \sqrt{1 - 4\epsilon^2}T_P(\mathbf{u}_i))$ , where  $T_P(\mathbf{v})$  is the tensoring of  $\mathbf{v}$  induced by the polynomial  $P$ . We fix the values of  $\alpha$  and  $\beta$  defining  $P$  (and hence, defining the vectors  $\mathbf{z}_i$ ) according to the following lemma implicit in [10]:

**Lemma 1** ([10]).  $\frac{2m}{\gamma} \geq \frac{1}{\gamma} \dots m \dots \epsilon > 3\gamma \dots \alpha, \beta > 0$

$$\alpha < 7.5\gamma,$$

$$2\beta + \alpha > \frac{4\epsilon}{1 + 2\epsilon} - 4\gamma,$$

$\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_n$  VERTEX COVER  
(II) □

A translated version of the vector set  $\{\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_n\}$  lay at root of the  $LS_+$  lower bounds proved in [10]. Specifically, the Gram matrix of the vectors  $\mathbf{v}_i = \frac{\mathbf{z}_0 + \mathbf{z}_i}{2}$  was shown to be a solution for the VERTEX COVER SDP resulting from  $O(\sqrt{\log n / \log \log n})$  rounds of  $LS_+$  lift-and-project.

The remainder of this section is devoted to proving the following theorem.

**Theorem 2.**  $\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_n \dots r \leq \frac{2}{45} \frac{\epsilon}{\gamma}$

We claim that Theorem II follows immediately from Theorem 2. Indeed, note first that the value of SDP (II) on the vectors  $\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_n$  is  $(1 + \epsilon)2^{m-1}$ . On the other hand, recall that the underlying graph  $G_m^\gamma$  has minimal vertex cover size  $(1 - o(1))2^m$  whenever  $\gamma = \Omega(\sqrt{\log m/m})$ . Hence, Theorem II follows by taking  $\epsilon > 0$  to be any arbitrarily small constant and  $\gamma = \Omega(\sqrt{\log m/m})$ .

As an aside, we note that our vectors  $\{\mathbf{z}_i\}$  also satisfy the “antipodal” triangle inequalities  $(\pm\mathbf{z}_i - \pm\mathbf{z}_j) \cdot (\pm\mathbf{z}_i - \pm\mathbf{z}_k) \geq 0$  for all  $i, j, k \in \{0\} \cup V$ . Recall that these inequalities define the SDP at root of Karakostas’s [14] VERTEX COVER algorithm. That our vectors satisfy these inequalities can be seen as follows. Consider the subset  $\{\mathbf{z}_i\}_{i \geq 1}$ . For each coordinate, the vectors in this subset take on at most 2 different values, and hence this subset is  $\ell_1$ -embeddable. Moreover, this remains true even if we replace some (or all) of the  $\mathbf{z}_i$  by  $-\mathbf{z}_i$ . Hence, it suffices to consider only the “antipodal” triangle inequalities involving  $\mathbf{z}_0$ . The validity of these inequalities then follows easily from the fact that the  $\mathbf{z}_i$  satisfy the standard triangle inequalities (by Lemma 1) and the fact that the value of  $\mathbf{z}_i \cdot \mathbf{z}_0$  does not depend on  $i$ .

Before giving the proof of Theorem 2 we give some intuition. Note that the vector set  $\{\mathbf{z}_i\}$  is the result of a perturbation applied to the following simple-minded  $\ell_1$  metric: Let  $D = \{\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_n\}$  be the metric obtained by taking  $\mathbf{v}_i$  to be the (normalized version of) the vectors of the  $m$ -dimensional cube, and let  $\mathbf{v}_0$  be a unit vector perpendicular to all  $\mathbf{v}_i$ . Notice that these vectors are precisely the vectors we would have obtained if we had used the polynomial  $P(x) = x$  to define the tensored vectors  $\mathbf{z}_i$  (corresponding to taking  $\epsilon = \gamma = 0$ ). The metric  $D$  is easily seen to be an  $\ell_1$  metric: take the Hamming cube and place the zeroth point at the origin to get an  $\ell_1$  embedding that is an isometry. Since  $D$  is  $\ell_1$ , every hypermetric inequality is valid for it. On the other hand,  $D$  does not satisfy even the basic conditions of SDP (1) (e.g., the edge constraints) with respect to our graph of interest (i.e.,  $G_m^\gamma$  with  $\gamma > 0$ ) and any basic attempts to remedy that will violate even the triangle inequality: A subtle way of perturbing  $D$  via the tensoring polynomial  $P$  will be required. By focusing on the pure hypermetrics, we can give some intuition about why our construction works and why the critical value of  $k$  is  $O(\epsilon/\gamma)$  (for non-pure hypermetrics, this intuition is less accurate). Given any choice of  $\alpha, \beta > 0$  we get a set of tensored vectors  $\mathbf{z}_i$  whose distances are a perturbation of  $D$  by an additive factor  $D_\Delta$ . As mentioned in the proof outline in the introduction and in light of Lemma 1, it suffices to restrict our attention only to inequalities supported on more than three points. Since any given pure hypermetric inequality defined by the  $b_i$ ’s must be satisfied by  $D$ , it is sufficient to prove that it is satisfied for the perturbed component of the metric, i.e.,  $D_\Delta$ . Analyzing this inequality on  $D_\Delta$  then shows that  $\sum_{i,j} b_i b_j d_{ij} \leq -2\epsilon + C\gamma k$ , where  $C$  is a universal constant and the  $d_{ij}$  are the distances defined by  $D_\Delta$ . Consequently, as long as  $k = O(\epsilon/\gamma)$ , the inequality holds for  $D_\Delta$ . Hence it holds for  $D + D_\Delta$ , the metric resulting from the  $\mathbf{z}_i$ ’s as well.

□ By Lemma 1 we already know that the vectors satisfy all hypermetric inequalities on three points, namely, the triangle inequalities.

So we only need to show that the solution satisfies hypermetric inequalities on 4 or more points. This is an important point since the arguments we will use to handle hypermetric inequalities on at least 4 points cannot be applied to the triangle inequalities.

Consider the set of vectors  $\{\mathbf{z}_i\}$ ,  $i \geq 1$ . For each coordinate, the vectors in this subset take on at most 2 different values, and hence this subset is  $\ell_1$ -embeddable.

Therefore, any  $\ell_1$  inequality (and in particular any hypermetric inequality) not involving  $\mathbf{z}_0$  must be satisfied.

Now let  $B = \mathbf{b}\mathbf{b}^t \in \mathcal{C}^*$ , where  $\mathbf{b} \in \mathbb{Z}^{k+1}$  and  $\sum_{i=0}^k b_i = 1$ , be any hypermetric inequality supported on  $r = k + 1$  points. By the above discussion, it suffices to consider the case where  $\mathbf{z}_0$  is one of the points, and we can assume that the points are  $0, 1, \dots, k$ . Our goal is to show that  $\sum_{i < j \leq k} B_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|^2 \leq 0$ . By definition, for  $i, j \geq 1$ ,

$$\|\mathbf{z}_i - \mathbf{z}_j\|^2 = 2 - 2(4\epsilon^2 + (1 - 4\epsilon^2)P(u_i \cdot u_j)) = 2(1 - 4\epsilon^2)(1 - P(u_i \cdot u_j)),$$

and  $\|\mathbf{z}_i - \mathbf{z}_0\|^2 = 2 - 4\epsilon$ . Hence,

$$\sum_{0 \leq i < j \leq k} B_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|^2 = 2(1 - 2\epsilon) \sum_{i=1}^k B_{0i} + 2(1 - 4\epsilon^2) \sum_{0 < i < j \leq k} B_{ij} (1 - P(u_i \cdot u_j)).$$

Therefore, we need to show

$$\sum_{i=1}^k B_{0i} + (1 + 2\epsilon) \sum_{0 < i < j \leq k} B_{ij} (1 - P(u_i \cdot u_j)) \leq 0. \tag{3}$$

We will require the technical lemma below, but first some definitions. By homogeneity we may assume  $b_0 < 0$  (and hence that  $b_0 \leq -1$  since  $b_0 \in \mathbb{Z}$ ). Let

$$S = \{i \in [k] : b_i > 0\}, \quad T = \{i \in [k] : b_i < 0\}.$$

Next let  $H_{ij} = (\mathbf{u}_i \cdot \mathbf{u}_j + 1)(\mathbf{u}_i \cdot \mathbf{u}_j)^{\frac{2m}{\gamma}}$  and  $M_{ij} = (\mathbf{u}_i \cdot \mathbf{u}_j)^{\frac{1}{\gamma}}$ , and let  $\Delta_{ij}$  be the Hamming distance between  $\mathbf{u}_i$  and  $\mathbf{u}_j$ . With these definitions we can then write  $P(\mathbf{u}_i \cdot \mathbf{u}_j) = \beta H_{ij} + \alpha M_{ij} + (1 - \alpha - 2\beta)(1 - \frac{2}{m}\Delta_{ij})$ .

**Lemma 2.**  $\dots \gamma \in \dots m, \dots \dots \dots \square \dots$

$$\begin{aligned} \sum_{0 < i < j \leq k} B_{ij} &= \frac{1}{2}((1 - b_0)^2 - \sum_{i=1}^k b_i^2) \\ \sum_{0 < i < j \leq k} B_{ij}(-\beta H_{ij} - \alpha M_{ij}) &\leq 15\gamma \sum_{i \in S, j \in T} b_i(-b_j) \\ \sum_{0 < i < j \leq k} B_{ij} \Delta_{ij} &\leq \frac{1}{4}m(1 - b_0)^2 \end{aligned}$$

The first equality is an immediate consequence of the fact that  $\sum_{i=1}^k b_i = 1 - b_0$  and that  $(\sum_{i=1}^k b_i)^2 = \sum_{i=1}^k b_i^2 + 2 \sum_{0 < i < j \leq k} b_i b_j$ .

For the second inequality, note first that  $\mathbf{u}_i \cdot \mathbf{u}_j \leq 1 - 1/m$ . Hence,  $H_{ij}$  is negligible for all  $i \neq j$ . Moreover, since the  $\mathbf{u}_i$  are unit vectors and  $1/\gamma$  is even, it follows that  $0 \leq M_{ij} \leq 1$ . Hence, by the bounds for  $\alpha$  and  $\beta$  given by Lemma  $\square$  it follows that  $\beta H_{ij} + \alpha M_{ij} \leq 15\gamma$  and the second inequality follows.

For the last inequality notice that since  $\Delta_{ij}$  is the sum of  $m$  cut metrics (defined by the  $m$  coordinates), it is enough to show that for every subset  $I \subset \{0, 1, \dots, k\}$ ,

$$\sum_{0 < i < j \leq k} B_{ij} \delta_I(i, j) \leq \frac{1}{4}(1 - b_0)^2.$$



Indeed, using the fact that  $B$  is a hypermetric we have,

$$\sum_{0 < i < j \leq k} B_{ij} \delta_I(i, j) = \sum_{i \in I, j \notin I} b_i b_j = \left( \sum_{i \in I} b_i \right) \cdot \left( 1 - b_0 - \sum_{i \in I} b_i \right) \leq \left( \frac{1 - b_0}{2} \right)^2.$$

□

We can now bound the left-hand-side of (3). To begin with, we have,

$$\begin{aligned} & \sum_{i=1}^k B_{0i} + (1 + 2\epsilon) \sum_{0 < i < j \leq k} B_{ij} (1 - P(u_i \cdot u_j)) \\ &= \sum_{i=1}^k B_{0i} + (1 + 2\epsilon) \sum_{0 < i < j \leq k} B_{ij} (1 - \beta H_{ij} - \alpha M_{ij} - (1 - \alpha - 2\beta)(1 - \frac{2}{m} \Delta_{ij})) \\ &= \sum_{i=1}^k B_{0i} + (1 + 2\epsilon) \sum_{0 < i < j \leq k} B_{ij} (-\beta H_{ij} - \alpha M_{ij} + \alpha + 2\beta + (1 - \alpha - 2\beta) \frac{2}{m} \Delta_{ij}). \end{aligned}$$

Applying the inequalities from Lemma 2 it then follows that the above is upper-bounded by

$$\begin{aligned} & b_0(1 - b_0) + (1 + 2\epsilon) \left( 15\gamma \sum_{i \in S, j \in T} b_i(-b_j) + \frac{1}{2}(\alpha + 2\beta) \left[ (1 - b_0)^2 - \sum_{i=1}^k b_i^2 \right] + \frac{1}{2}(1 - \alpha - 2\beta)(1 - b_0)^2 \right) \\ &= \frac{1}{2}(1 - b_0^2) + 2\epsilon \frac{1}{2}(1 - b_0)^2 + (1 + 2\epsilon) \left( 15\gamma \sum_{i \in S, j \in T} b_i(-b_j) - \frac{1}{2}(\alpha + 2\beta) \sum_{i=1}^k b_i^2 \right) \\ &< \frac{1}{2}(1 - b_0^2) + 2\epsilon \frac{1}{2}(1 - b_0)^2 + (1 + 2\epsilon) \left( 15\gamma \sum_{i \in S, j \in T} b_i(-b_j) - \left[ \frac{2\epsilon}{1 + 2\epsilon} - 2\gamma \right] \sum_{i=1}^k b_i^2 \right) \\ &= \frac{1}{2}(1 - b_0^2) + 2\epsilon \frac{1}{2}(1 - b_0)^2 - 2\epsilon \sum_{i=1}^k b_i^2 + (1 + 2\epsilon) \left( 15\gamma \sum_{i \in S, j \in T} b_i(-b_j) + 2\gamma \sum_{i=1}^k b_i^2 \right) \\ &< \frac{1}{2}(1 - b_0^2) - \epsilon \left( 2 \sum_{i=1}^k b_i^2 - (1 - b_0)^2 \right) + 15\gamma(1 + 2\epsilon) \left( \sum_{i \in S, j \in T} b_i(-b_j) + \sum_{i=1}^k b_i^2 \right) \\ &< \frac{1}{2}(1 - b_0^2) - \epsilon \left( 2 \sum_{i=1}^k b_i^2 - (1 - b_0)^2 \right) + 30\gamma \left( \sum_{i \in S, j \in T} b_i(-b_j) + \sum_{i=1}^k b_i^2 \right). \end{aligned}$$

Note that since the hypermetric inequality we are considering is a triangle inequality, it follows that we must have  $\sum_{i>0} b_i^2 \geq 3$ . But then, the following technical lemma can be used to show that the above is bounded by 0, completing the proof of the theorem.

**Lemma 3.** . . .  $k \leq \frac{2}{45} \frac{\epsilon}{\gamma} - 1$  . . .  $0 < \epsilon < \frac{1}{6}$  . . .  $\gamma > 0$  . . .  $b_0 \leq -1$    
 $\sum b_i^2 \geq 3$  . . .  $b_i \neq 0$  . . .  $i = 1, \dots, k$

$$\frac{1}{2}(1 - b_0^2) - 2\epsilon \left( \sum_{i=1}^k b_i^2 - \frac{1}{2}(1 - b_0)^2 \right) + 30\gamma \left( \sum_{i \in S, j \in T} b_i(-b_j) + \sum_{i=1}^k b_i^2 \right) < 0.$$

It is not hard to check that since  $\epsilon < \frac{1}{6}$  and  $b_0$  is a (strictly) negative integer, we have

$$\frac{1}{2}(1 - b_0^2) - 2\epsilon \left( \sum_{i=1}^k b_i^2 - \frac{1}{2}(1 - b_0)^2 \right) \leq -2\epsilon \left( \sum_{i=1}^k b_i^2 - 2 \right) \leq -\frac{2\epsilon}{3} \sum_{i=1}^k b_i^2 .$$

Note that it was critical to have  $\sum_{i>0} b_i^2 \geq 3$  here, as only then can we claim that  $\sum_{i=1}^k b_i^2 - 2$  is a positive constant. Indeed, for the triangle inequality ( $b_0 = -1, b_1 = b_2 = 1$ ), i.e., the only hypermetric inequality for which this doesn't hold, we cannot expect any method bounding the slack of the inequality to work: the VERTEX COVER edge constraints force the triangle inequality to be tight for edges!

It now suffices to prove that

$$-\frac{2\epsilon}{3} \sum_{i=1}^k b_i^2 + 30\gamma \left( \sum_{i \in S, j \in T} b_i(-b_j) + \sum_{i=1}^k b_i^2 \right) < 0. \tag{4}$$

Let  $s, t$  be the cardinalities of  $S, T$ , respectively, and let  $x = \sum_{i \in S} b_i$  and  $y = \sum_{i \in T} (-b_i)$ . Now, using the Cauchy-Schwartz inequality and the fact that  $s, t \leq k$ , we get

$$\frac{\sum_{i \in S, j \in T} b_i(-b_j) + \sum_{i=1}^k b_i^2}{\sum_{i=1}^k b_i^2} \leq 1 + \frac{xy}{s(x/s)^2 + t(y/t)^2} \leq 1 + k \frac{xy}{x^2 + y^2} \leq 1 + k/2.$$

(Note that if  $y = t = 0$  the bound is trivial and we therefore ignored this case above.) Hence,

$$-\frac{2\epsilon}{3} \sum_{i=1}^k b_i^2 + 30\gamma \left( \sum_{i \in S, j \in T} b_i(-b_j) + \sum_{i=1}^k b_i^2 \right) < \left( -\frac{2\epsilon}{3} + 30\gamma(1 + k/2) \right) \sum_{i=1}^k b_i^2,$$

and so (4) holds as long as  $k \leq \frac{2}{45} \frac{\epsilon}{\gamma} - 1$ . □

Theorem 2 now follows. □

## 4 Hypermetric Inequalities vs. Lovász-Schrijver SDP Lift-and-Project

In this section we show that hypermetric inequalities need not be derived by Lovász and Schrijver's  $LS_+$  lift-and-project system. Our plan of attack is as follows. After stating all necessary definitions, we will first show that no pure hypermetric inequalities are derived by  $LS_+$  for the convex cone defined by the inequalities  $0 \leq x_i \leq x_0, i = 1, \dots, n$ . We will then use this result to show the following for VERTEX COVER: Fix a graph  $G$  and an independent set  $S$  in  $G$ , and consider a VERTEX COVER SDP for  $G$  derived using  $LS_+$  lift-and-project. Then

the constraints defining this SDP do not imply any of the pure hypermetric constraints supported on  $S$ .

We begin by defining the Lovász-Schrijver  $LS_+$  lift-and-project system [18]. In what follows all vectors will be indexed starting at 0. Recall that a set  $C \subset \mathbb{R}^n$  is a convex cone if for every  $\mathbf{y}, \mathbf{z} \in C$  and for every  $\alpha, \beta \geq 0$ ,  $\alpha\mathbf{y} + \beta\mathbf{z} \in C$ . Given a convex cone  $C \subset \mathbb{R}^{n+1}$  we denote its projection onto the hyperplane  $x_0 = 1$  by  $C|_{x_0=1}$ . Let  $\mathbf{e}_i$  denote the vector with 1 in coordinate  $i$  and 0 everywhere else. Let  $Q_n \subset \mathbb{R}^{n+1}$  be the convex cone defined by the constraints  $0 \leq x_i \leq x_0$  and fix a convex cone  $C \subset Q_n$ . The lifted cone  $M_+(C) \subseteq \mathbb{R}^{(n+1) \times (n+1)}$  consists of all positive semidefinite matrices  $(n+1) \times (n+1)$  matrices  $Y$  such that,

Property I. For all  $i = 0, 1, \dots, n$ ,  $Y_{0i} = Y_{ii}$ .

Property II. For all  $i = 0, 1, \dots, n$ ,  $Y\mathbf{e}_i, Y\mathbf{e}_0 - Y\mathbf{e}_i \in C$ .

The cone  $M_+(C)$  is the  $LS_+$  lift-and-project system for  $C$ . This procedure can be iterated by projecting  $M_+(C)$  back to  $\mathbb{R}^{n+1}$  and then re-applying the  $M_+$  operator to the projection. In particular, let  $N_+(C) = \{Y\mathbf{e}_0 : Y \in M_+(C)\} \subseteq \mathbb{R}^{n+1}$ . Define  $N_+^k(C)$  inductively by setting  $N_+^0(C) = C$  and  $N_+^k(C) = N_+(N_+^{k-1}(C))$ , and define  $M_+^k(C)$  to be  $M_+(N_+^{k-1}(C))$ . Lovász and Schrijver show that  $N_+^{k+1}(C) \subseteq N_+^k(C)$  and  $M_+^{k+1}(C) \subseteq M_+^k(C)$  and that moreover these containment are proper if and only if  $N_+^k(C)|_{x_0=1}$  is not the integral hull of  $C|_{x_0=1}$ . Moreover, they show that  $N_+^n(C)|_{x_0=1}$  is equal to the integral hull of  $C|_{x_0=1}$ . It is useful to note, that  $Y \in M_+^k(C) \subseteq \mathbb{R}^{(n+1) \times (n+1)}$  if and only if  $Y$  is PSD and satisfies both Property I and the following property:

Property II'. For all  $i = 0, 1, \dots, n$ ,  $Y\mathbf{e}_i, Y\mathbf{e}_0 - Y\mathbf{e}_i \in N_+^{k-1}(C)$ .

With these definitions in hand, we can now begin by showing that  $M_+(Q_n)$  does not satisfy any pure hypermetric constraint (recall that  $Q_n$  is the cone satisfying  $0 \leq x_i \leq x_0$  for all  $i = 1, \dots, n$ ). As a warm up we examine the triangle inequality of SDP (1) for a three vertex graph with no edges. Note that this SDP has no edge constraints. Moreover, any vector solution  $\mathbf{v}_i$  can be mapped using the affine transformation  $\mathbf{v}_i \rightarrow (\mathbf{v}_i + \mathbf{v}_0)/2$  to a set of vectors whose Gram matrix is in  $M_+(Q_3)$ , and vice versa. Now consider three vectors  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$  that correspond to the three vertices of the graph. Geometrically it is possible to place these vectors such that the Gram matrix of  $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$  satisfies Properties I and II above for an  $LS_+$  tightening, yet  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$  violate triangle inequality. We can accomplish this by making  $\mathbf{v}_1$  and  $\mathbf{v}_2$  almost coincide and placing  $\mathbf{v}_3$  between them.

Our counterexample for hypermetrics will be a generalization of the following more subtle matrix in  $M_+(Q_3)$  violating triangle inequality:

$$Y = \begin{pmatrix} 1 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & 0 & \beta\epsilon \\ \epsilon & 0 & \epsilon & \beta\epsilon \\ \epsilon & \beta\epsilon & \beta\epsilon & \epsilon \end{pmatrix}.$$

By having  $\epsilon \in (0, 1/2)$  and  $\beta \in [0, 1]$  we ensure  $Y$  satisfies Properties I and II. One can show that by setting  $\epsilon$  arbitrarily close to 0 and  $\beta$  close to but bigger

than  $1/2$ , we ensure that  $Y$  is PSD, while ensuring that its Cholesky decomposition violates the triangle inequality. This matrix sacrifices some of the above geometric intuition to make our calculations easier.

This construction can be extended to show that  $M_+(Q_n)$  does not satisfy any inequality  $\sum b_i b_j d_{ij} \leq 0$  where  $b$  is a vector of length  $n = 2k + 1$ ,  $\sum b_i = 1$ , and for all  $i$ ,  $|b_i| = 1$ . Indeed, consider an inequality on  $2k + 1$  points defined by the vector  $(0, b_1, b_2, \dots, b_{2k+1}) \in \mathbb{Z}_+^{2k+2}$  (note that  $b_0 = 0$ ) where  $b_i = 1$  for  $i = 1, \dots, k + 1$  and  $b_i = -1$  for  $i = k + 2, \dots, 2k + 2$ . In this way we naturally split the points into two clusters of size  $k + 1$  and  $k$  points. The associated inequality requires that the sum of distances across the clusters dominates the sum of distances within the clusters. Define the distance within the clusters as  $2\epsilon$ , and the distance across the clusters as  $2\epsilon(1 - \beta)$ . We have  $k(k + 1)$  cross pairs and  $\binom{k}{2} + \binom{k+1}{2} = k^2$  inner pairs. Therefore in order to violate the inequality, we should have  $2\epsilon(1 - \beta)k(k + 1) < 2\epsilon k^2$ . In other words it suffices for  $\beta$  to be slightly bigger than  $\frac{1}{k+1}$  (this will be crucial later).

Define the matrix

$$Y^{(s,t)} = \begin{pmatrix} 1 & \epsilon J_{1,s} & \epsilon J_{1,t} \\ \epsilon J_{s,1} & \epsilon I_s & \epsilon \beta J_{s,t} \\ \epsilon J_{t,1} & \epsilon \beta J_{t,s} & \epsilon I_t \end{pmatrix},$$

where  $J_{m,n}$  is the  $m \times n$  all-1 matrix, and  $I_n$  is the  $n \times n$  identity matrix (note that  $s = 2, t = 1$  gives  $Y$  above). The configuration described above can be realized by the matrix  $Y^{(k+1,k)}$  of order  $(2k + 2)$ . Similarly as in the case of the triangle inequality,  $Y^{(s,t)}$  satisfies Properties I and II as long as  $\epsilon \in (0, 1/2)$  and  $b \in [0, 1]$ .

Hence,  $Y^{(k+1,k)}$  is in  $M_+(Q_n)$  provided we can show that it is PSD. This is implied by the following technical lemma.

**Lemma 4.** *Let  $s, t \in \mathbb{N}$  such that  $s + t = 2k + 1$ . Let  $\epsilon \in (0, 1/2)$  and  $\beta > \frac{1}{k+1}$ . Then  $Y^{(s,t)} \in \mathbb{R}^{(2k+2) \times (2k+2)}$  is PSD.*

To simplify notation, we will denote  $\frac{1}{\epsilon} Y^{(s,t,\epsilon,\beta)}$  by  $Y^{(s,t)}$ .

We begin by computing all principal minors of  $Y^{(s,t)}$ . Subtracting the third row from the second in  $Y^{(s,t)}$ , we get  $\det(Y^{(s,t)}) = \det(Y^{(s-1,t)}) + \det(L^{(s-1,t)})$ , where  $L^{(s,t)}$  is the same matrix as  $Y^{(s,t)}$  except that  $L_{22}^{(s,t)} = 0$  (instead of 1).

The same operation on rows shows that  $\det(L^{(s,t)}) = \det(L^{(1,t)})$ . Next denote by  $K^{(1,t)}$  the same matrix as  $L^{(1,t)}$  except that  $K_{33}^{(1,t)} = 0$  (instead of 1). Subtracting the fourth row from the third in  $L^{(1,t)}$  we get  $\det(L^{(1,t)}) = \det(L^{(1,t-1)}) + \det(K^{(1,t-1)})$  where again  $\det(K^{(1,t)}) = \det(K^{(1,1)})$ . Finally let  $M^{(1,t)}$  be the same matrix as  $Y^{(1,t)}$  except that  $M_{33}^{(1,t)} = 0$  (instead of 1). Again, the same row operation in  $Y^{(1,t)}$  gives  $\det(Y^{(1,t)}) = \det(Y^{(1,t-1)}) + \det(M^{(1,t-1)})$  with  $\det(M^{(1,t)}) = \det(M^{(1,1)})$ .

For simplicity denote  $Y^{(1,1)}, L^{(1,1)}, M^{(1,1)}, K^{(1,1)}$  by  $Y, L, M, K$  respectively. Then for these base matrices

$$Y = \begin{pmatrix} 1/\epsilon & 1 & 1 \\ 1 & 1 & \beta \\ 1 & \beta & 1 \end{pmatrix} \quad L = \begin{pmatrix} 1/\epsilon & 1 & 1 \\ 1 & 0 & \beta \\ 1 & \beta & 1 \end{pmatrix} \quad M = \begin{pmatrix} 1/\epsilon & 1 & 1 \\ 1 & 1 & \beta \\ 1 & \beta & 0 \end{pmatrix} \quad K = \begin{pmatrix} 1/\epsilon & 1 & 1 \\ 1 & 0 & \beta \\ 1 & \beta & 0 \end{pmatrix}$$

we have

$$\begin{aligned} \det(Y) &= \frac{1}{\epsilon}(1 - \beta^2 + 2\beta\epsilon - 2\epsilon) \\ \det(L) &= \det(M) = \frac{1}{\epsilon}(-\beta^2 + 2\beta\epsilon - \epsilon) \\ \det(K) &= \frac{\beta}{\epsilon}(-\beta^2 + 2\epsilon). \end{aligned}$$

Using these values, we have

$$\begin{aligned} \det(Y^{(s,t)}) &= \det(Y^{(1,t)}) + (s - 1)(\det(L) + (t - 1)\det(K)) \\ &= \det(Y) + (t - 1)\det(M) + (s - 1)(\det(L) + (t - 1)\det(K)) \\ &= \frac{1}{\epsilon}(1 - st\beta^2 + \epsilon(-t - s + 2st\beta)). \end{aligned}$$

Recall that we required  $\beta > \frac{1}{k+1}$  and so we can take  $\beta$  arbitrarily close to that bound. But then

$$st\beta^2 \leq \left(\frac{2k+1}{2}\right)^2 \frac{1}{(k+1)^2} = \left(\frac{2k+1}{2k+2}\right)^2 < 1$$

making  $\det(Y^{(s,t)})$  strictly positive for sufficient small  $\epsilon$ . □

We are ready now to show that VERTEX COVER SDPs in the  $LS_+$  hierarchy violate pure hypermetrics on any independent set. Fix an  $n$ -vertex graph  $G = (V, E)$  and consider the convex cone  $C \subset Q_n$  consisting of all vectors  $x \in \mathbb{R}^{n+1}$  such that  $x_i + x_j \geq x_0$ . Then  $LS_+$  lifting yields the following sequence of SDPs for  $G$ :  $M_+(C), M_+^2(C), \dots$ . We will show that for all  $k$ , every independent set  $S$  in  $G$ , and all pure hypermetrics  $B$  supported on  $S$ , there exists  $Y \in M_+^k(C)$  such that  $Y$  does not satisfy  $B$ .

To that end, fix  $k$  and  $S$ , and let  $s = |S|$  be odd. Without loss of generality, assume that  $S = \{1, 2, \dots, s\}$ . Fix a pure hypermetric  $B$  defined on the set  $S$ . By the discussion above we know that there exists  $Y' \in M_+(Q_s)$  that violates the pure hypermetric  $B$ . Let  $\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_s$  be the Cholesky decomposition for  $Y'$ . Now let  $Y \in \mathbb{R}^{(n+1) \times (n+1)}$  be the matrix with Cholesky decomposition  $\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_s, \mathbf{v}'_{s+1}, \dots, \mathbf{v}'_n$  where  $\mathbf{v}'_j = \mathbf{v}_0$  for all  $j \geq s + 1$ . By construction  $Y$  is PSD, satisfies Property I, and does not satisfy  $B$  on  $S$ . So it suffices to verify Property II' in order to show that  $Y \in M_+^k(C)$ . Note that  $Y\mathbf{e}_i$  is the all-1 vector for all  $i \geq s + 1$  and hence Property II' holds for all  $i \geq s + 1$  since the all-1 vector is in the integral hull and hence in  $N_+^k(C)$  for all  $k$ . Now consider a vector  $Y\mathbf{e}_i$  where  $1 \leq i \leq s$ . Note that  $Y_{00} = Y_{0j}$  for all  $j \geq s + 1$ . But then, since  $S$  is independent, it follows that the projection of  $Y\mathbf{e}_i$  onto the hyperplane  $x_0 = 1$  is also in the integral hull and hence in  $N_+^k(C)$ . Similarly, it follows that

$Y(\mathbf{e}_0 - \mathbf{e}_i)$  is also in  $N_+^k(C)$  whenever  $1 \leq i \leq s$ . So Property II' holds for all  $i$ , and  $Y \in M_+^k(C)$ .

We end this section by remarking that the above arguments can be combined with those from [10] to show that there is a graph  $G$  for which  $O(\sqrt{\log n / \log \log n})$  rounds of  $LS_+$  produce an SDP which (a) does not satisfy the triangle inequality, (b) has integrality gap  $2 - o(1)$ . The argument, which we do not have room to go into here, considers the Frankl-Rödl graph  $G_m^\gamma$  to which we append three isolated vertices. The idea is to not satisfy the triangle inequality on the isolated vertices while the remaining vertices will essentially employ the SDP solutions from [10].

## Acknowledgements

We thank Toniann Pitassi for many helpful discussions. The first author would also like to thank Periklis Papakonstantinou for his help on some technical aspects of this work.

## References

1. Arora, S., Lovász, L., Newman, I., Rabani, Y., Rabinovich, Y., Vempala, S.: Local versus global properties of metric spaces. In: SODA, pp. 41–50. ACM Press, New York (2006)
2. Arora, S., Rao, S., Vazirani, U.: Expander flows, geometric embeddings and graph partitioning. In: Proceedings of the 36th Annual ACM Symposium on Theory of Computing (electronic), pp. 222–231. ACM, New York (2004)
3. Avis, D., Umemoto, J.: Stronger linear programming relaxations of max-cut. *Mathematical Programming* 97(3), 451–469 (2003)
4. Charikar, M.: On semidefinite programming relaxations for graph coloring and vertex cover. In: SODA 2002. Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, pp. 616–620 (2002); Society for Industrial and Applied Mathematics
5. Charikar, M., Makarychev, K., Makarychev, Y.: Local global tradeoffs in metric embeddings. In: Proceedings of the Forty-Eighth Annual IEEE Symposium on Foundations of Computer Science, pp. 713–723. IEEE, Los Alamitos (2007)
6. Devanur, N.R., Khot, S.A., Saket, R., Vishnoi, N.K.: Integrality gaps for sparsest cut and minimum linear arrangement problems. In: STOC 2006. Proceedings of the 38th Annual ACM Symposium on Theory of Computing, pp. 537–546. ACM, New York (2006)
7. Deza, M., Laurent, M.: *Geometry of cuts and metrics*. Algorithms and Combinatorics, vol. 15. Springer, Berlin, Heidelberg, New York, Barcelona, Budapest, Hong Kong, London, Mailand, Paris, Santa Clara, Singapur, Tokyo (1997)
8. Dinur, I., Safra, S.: On the hardness of approximating minimum vertex-cover. *Annals of Mathematics* 162(1), 439–486 (2005)
9. Frankl, P., Rödl, V.: Forbidden intersections. *Trans. Amer. Math. Soc.* 300(1), 259–286 (1987)

10. Georgiou, K., Magen, A., Pitassi, T., Toulakis, I.: Integrality gaps of  $2 - o(1)$  for vertex cover SDPs in the Lovász-Schrijver hierarchy. In: Proceedings of the Forty-Eighth Annual IEEE Symposium on Foundations of Computer Science, pp. 702–712. IEEE, Los Alamitos (2007)
11. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.* 42(6), 1115–1145 (1995)
12. Halperin, E.: Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM J. Comput(electronic)* 31(5), 1608–1623 (2002)
13. Hatami, H., Magen, A., Markakis, E.: Integrality gaps of semidefinite programs for vertex cover and relations to 1 embeddability of negative type metrics. In: APPROX-RANDOM, pp. 164–179 (2007)
14. Karakostas, G.: A better approximation ratio for the vertex cover problem. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1043–1050. Springer, Heidelberg (2005)
15. Khot, S.: On the power of unique 2-prover 1-round games. In: Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing, pp. 767–775. ACM, New York (2002)
16. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within  $2 - \epsilon$ . In: Proceedings of the 18th IEEE Conference on Computational Complexity, pp. 379–386 (2003)
17. Kleinberg, J., Goemans, M.X.: The Lovász theta function and a semidefinite programming relaxation of vertex cover. *SIAM J. Discrete Math.* 11(2), 196–204 (1998)
18. Lovász, L., Schrijver, A.: Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization* 1(2), 166–190 (1991)

# Tight Bounds for Permutation Flow Shop Scheduling

Viswanath Nagarajan<sup>1,\*</sup> and Maxim Sviridenko<sup>2</sup>

<sup>1</sup> Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213  
viswa@cmu.edu

<sup>2</sup> IBM T.J. Watson Research center, Yorktown Heights, NY 10598  
sviri@us.ibm.com

**Abstract.** In flow shop scheduling there are  $m$  machines and  $n$  jobs, such that every job has to be processed on the machines in the fixed order  $1, \dots, m$ . In the *permutation flow shop* problem, it is also required that each machine processes the set of all jobs in the *same* order. Formally, given  $n$  jobs along with their processing times on each machine, the goal is to compute a single permutation of the jobs  $\sigma : [n] \rightarrow [n]$ , that minimizes the maximum job completion time (makespan) of the schedule resulting from  $\sigma$ . The previously best known approximation guarantee for this problem was  $O(\sqrt{m \log m})$  [29]. In this paper, we obtain an improved  $O(\min\{\sqrt{m}, \sqrt{n}\})$  approximation algorithm for the permutation flow shop scheduling problem, by finding a connection between the scheduling problem and the longest increasing subsequence problem. Our approximation ratio is relative to the lower bounds of maximum job length and maximum machine load, and is the best possible such result. This also resolves an open question from [21], by algorithmically matching the gap between permutation and non-permutation schedules. We also consider the weighted completion time objective for the permutation flow shop scheduling problem. Using a natural linear programming relaxation, and our algorithm for the makespan objective, we obtain an  $O(\min\{\sqrt{m}, \sqrt{n}\})$  approximation algorithm for minimizing the total weighted completion time, improving upon the previously best known guarantee of  $\epsilon m$  for any constant  $\epsilon > 0$  [30]. We give a matching lower bound on the integrality gap of our linear programming relaxation.

## 1 Introduction

In the *permutation flow shop* problem, there are  $m$  machines located in a fixed order (say, 1 through  $m$ ), and  $n$  jobs each of which consists of a sequence of operations on machines (in the order 1 through  $m$ ). For any job  $j \in \{1, \dots, n\}$  and machine  $i \in \{1, \dots, m\}$  the length of the operation of job  $j$  on machine  $i$  is called its *processing time*  $p_{ij}$ . A schedule for the jobs is feasible if (i) each machine processes at most one job at any time; (ii) for each job, its operations on the machines are

---

\* Supported in part by NSF grant CCF-0728841. Work done while visiting IBM T.J. Watson Research center.



processed in the fixed order 1 through  $m$ ; and (iii) each operation (of a job on a machine) is processed without interruption. The flow shop problem is a special case of permutation scheduling [5,6], which in turn is a special case of the general scheduling [3,15].

We study the permutation flow shop problem, which is the flow shop problem with the additional constraint that each machine processes all the jobs in the same order. So any feasible schedule to the permutation flow shop problem corresponds to a permutation of the  $n$  jobs. It is well-known [4] that there exists an optimal schedule for the ordinary flow shop problem having the same processing order (of jobs) on the first two machines and the same processing order on the last two machines. So the permutation flow shop problem is equivalent to the ordinary flow shop problem for  $m \leq 3$  machines. However, it is easy to construct an instance with  $m = 4$  machines where no permutation schedule is optimal for the ordinary flow shop problem.

Two natural objective functions that are typically considered for scheduling problems are makespan and weighted completion time. The makespan of a schedule is the completion time of its last job, i.e. maximum completion time among all jobs. For the weighted completion time objective, each job  $j$  comes with a weight  $w_j \geq 0$ , and the weighted completion time of a schedule is the weighted sum of completion times over all jobs.

## 1.1 Related Work

When the number of machines  $m$  is a fixed constant, a PTAS is known for the job-shop scheduling problem with the makespan objective due to Jansen et al. [12] and the total weighted completion time objective due to Fishkin et al. [7]. It seems quite likely, that similar techniques yield PTASes for the permutation flow shop scheduling problems with corresponding objective functions, although such results did not appear in the literature. For the ordinary flow shop problem with the makespan objective, the best known approximation guarantee is  $O(\log m (\log \log m)^{1+\epsilon})$ , where  $\epsilon > 0$  is any constant, due to Czumaj and Scheideler [5]; in fact this result holds for the more general class of acyclic-shop scheduling. Using the general result from [23] one can derive an approximation algorithm with the analogous performance guarantee for the flow shop scheduling problem with the total weighted completion time objective.

The following are two obvious lower bounds for the permutation flow shop scheduling problem with the makespan objective: maximum job length  $l = \max_{j=1}^n \{\sum_{i=1}^m p_{i,j}\}$ , and maximum machine load  $L = \max_{i=1}^m \{\sum_{j=1}^n p_{i,j}\}$ . Potts et al. [21] have shown a family of instances where the optimal makespan is  $\Omega(\sqrt{\min\{m, n\}})$  times the trivial lower bound  $(\max\{L, l\})$ . It was an open question in [21] to determine whether this bound is tight. The previously best known approximation guarantee for the makespan problem is  $O(\sqrt{m \log m})$  due to Sviridenko [29]; this guarantee is relative to the trivial lower bound. Prior to this, a number of algorithms [25,19,18] were shown to have a (tight) worst case guarantee of  $\lceil \frac{m}{2} \rceil$ . There are also some papers dealing with additive guarantees for this problem [26,29]. Sevastianov [26] gave an algorithm that always produces

a schedule of length at most  $L + O(m^2) \max_{i,j} p_{i,j}$ . Sviridenko [29] obtained a similar guarantee of  $(1 + \delta)L + K_\delta(m \log m) \max_{i,j} p_{i,j}$  for any  $\delta > 0$  (here  $K_\delta$  is a function depending on  $\delta$  alone). The best multiplicative approximation guarantee obtainable from these results is the  $O(\sqrt{m \log m})$  [29].

Smutnicki [30] gave a worst case analysis of several algorithms for the permutation flow shop problem with the weighted completion time objective. Assuming a  $\rho_k$  approximation guarantee for the problem on  $k$  machines, [30] gave an  $\frac{m}{k} \rho_k$  approximation algorithm for the problem on  $m$  machines. Assuming that there exists a PTAS for the permutation flow shop scheduling problem with the weighted completion time objective and fixed number of machines, one could obtain an  $\epsilon m$  guarantee for minimizing weighted completion time (for any constant  $\epsilon > 0$ ). Alternatively, we could use the  $(2 + \epsilon)$ -approximation algorithm from [28] that works basically for any shop scheduling problem with makespan criteria on constant number of machines. Otherwise, the best known guarantee is  $m$ . To the best of our knowledge this is the previously best known result for this problem.

The permutation flow shop scheduling problem has been very popular in the Operations Research community in the last 40 years and there is a significant body of work devoted to the design of heuristics for this problem. The survey paper [8] establishes a common framework for these heuristics, and describes main classes of algorithms and research directions.

## 1.2 Our Results and Paper Outline

We give a simple randomized algorithm (Section 2) for minimizing makespan that achieves an approximation guarantee of  $2\sqrt{\min\{m, n\}}$ . This guarantee is relative to the trivial lower bounds. The analysis is based on the connection between the permutation flow shop scheduling problem and the longest increasing subsequence problem. This connection is new and might be interesting in its own right. It also allows us to apply non-trivial probabilistic results on the expected value and concentration of the longest increasing subsequence in a random sequence [16,31,9].

Hence we answer the open question in Potts et al. [21], by matching algorithmically the  $\Omega(\sqrt{\min\{m, n\}})$  gap shown in [21]. We also show how this algorithm can be derandomized to obtain a deterministic approximation guarantee of  $3\sqrt{\min\{m, n\}}$ . This algorithm uses the derandomization technique of pessimistic estimators due to Raghavan [24] and certain ideas from the proof of concentration bound from [9]; the details are non-trivial and appear in Section 3. We note that among algorithms that are based on the trivial lower bounds, our algorithm is the best possible (up to a  $2\sqrt{2}$  factor).

On the first sight our improvement of  $O(\sqrt{\log m})$  upon the previously best known bound from [29] looks insignificant, but the proof in [29] is based on Chernoff Bounds and it is quite well-known that improving upon Chernoff Bounds based proofs requires substantially new insights on the problem structure. For example, for the famous combinatorial discrepancy problem the straightforward randomized algorithm yields the bound of  $O(\sqrt{n \log n})$ ; but using more

sophisticated techniques based on entropy and using a pigeon-hole principle, Spencer proved a better non-constructive bound of  $O(\sqrt{n})$  [2]. It is one of the well-known open questions to obtain a constructive (algorithmic) proof of Spencer's result; it is also unknown if Spencer's guarantee holds in the case when the matrix entries are arbitrary real numbers in the interval  $[0, 1]$  while Chernoff bounds can be easily applied even in this more general case. The key to our result is a decomposition of the matrix of processing times into a sum of permutation matrices and noticing that the "intersection" of each such matrix with any critical path corresponds to an increasing subsequence in a certain permutation.

Our second contribution is a partial explanation of great practical performance of the simple greedy algorithm for the permutation flow shop problem with the makespan objective. This algorithm was first suggested by Nawaz, Enscore and Ham [17] and is also known under the name of "insertion heuristic". This algorithm initially orders jobs in the decreasing order of the job lengths and inserts them one by one into the current schedule, always choosing the best position for a job in the current permutation. Although this algorithm is very simple and has superb practical performance [8], there is no analytical explanation of this phenomenon. Many practical algorithms either directly use the insertion heuristic or generalize it in some way. The natural way of analyzing such a heuristic would be to define a potential function which is improved on every step of the greedy procedure and prove some bound relating this function with the makespan. Although we were not able to apply this method to the insertion heuristic, our derandomization algorithm follows this framework. Moreover, our final derandomized algorithm resembles the greedy algorithm of Nawaz, Enscore and Ham [17]. The difference is that our algorithm greedily fixes the first few positions in the current permutation with respect to a certain potential function (derived from the concentration bound on length of the longest increasing subsequence [9]), while the greedy algorithm [17] just fixes a relative ordering of the first few jobs allowing unscheduled jobs to be scheduled in between later on.

We then consider the weighted completion time objective (Section 4) and use our algorithm for minimizing makespan to obtain an  $O(\sqrt{\min\{m, n\}})$  approximation algorithm for this problem. This algorithm uses the linear relaxation of a natural integer programming formulation for the problem. Our rounding algorithm is similar to the approach used in Queranne and Sviridenko [23] (and many other papers on scheduling with the weighted completion time objective [11, 11, 10]); the difference is that we need to ensure that when we apply the approach of geometric partitioning of the time interval, the schedule for each such interval satisfies the permutation constraint. We also show a matching  $\Omega(\sqrt{\min\{m, n\}})$  lower bound on the integrality gap of our LP relaxation.

### 1.3 Preliminaries

An instance of the permutation flow shop problem with  $m$  machines and  $n$  jobs is given by an  $m \times n$  matrix  $P = \{p_{i,j} \mid i = 1, \dots, m, j = 1, \dots, n\}$  of processing times, where  $p_{i,j}$  is the processing time of job  $j$  on machine  $i$ . We often denote

the set  $\{1, \dots, n\}$  of all jobs by  $[n]$ , and the set  $\{1, \dots, m\}$  of all machines by  $[m]$ . Any feasible schedule for permutation flow shop corresponds to a permutation of the  $n$  jobs. Given a permutation  $\pi : [n] \rightarrow [n]$  of jobs, the complete schedule of job-operations on machines can be obtained by running jobs on machines in the order  $\pi$  while maintaining the minimum possible wait-time between operations. It is convenient to think of  $\pi$  as a mapping from the set of  $n$  possible positions to the set of  $n$  jobs. Therefore,  $\pi(p)$  denotes the job located at position  $p$ . For any permutation  $\pi$  of the  $n$  jobs and a job  $j \in [n]$ , we use  $C_j^\pi$  to denote the completion time of job  $j$  under the schedule  $\pi$ ; we also denote the makespan of schedule  $\pi$  by  $C_{max}^\pi = \max_{j=1}^n C_j^\pi$ . Given non-negative weights  $\{w_j\}_{j=1}^n$  for the jobs, the weighted completion time objective of the schedule corresponding to permutation  $\pi$  is  $\sum_{j=1}^n w_j C_j^\pi$ .

A sequence  $\langle (x_1, y_1), \dots, (x_t, y_t) \rangle$  of matrix positions such that  $(x_1, y_1) = (1, 1)$ ,  $(x_t, y_t) = (m, n)$ , and for each  $1 \leq i < t$  either  $x_{i+1} = x_i + 1$  &  $y_{i+1} = y_i$  or  $x_{i+1} = x_i$  &  $y_{i+1} = y_i + 1$ . In particular, this definition implies that each monotone path in an  $m \times n$  matrix consists of exactly  $t = m + n - 1$  positions. We denote the set of all monotone paths in an  $m \times n$  matrix by  $\mathcal{M}_{m,n}$ .

A map  $\tau : [n] \rightarrow X \cup \{\phi\}$  where  $X \subseteq [m]$  is called a partial permutation if there is a subset  $Y \subseteq [n]$  with  $|Y| = |X|$  such that (i)  $\tau(Y) = X$  ( $\tau$  is a one-to-one map from  $Y$  to  $X$ ); and (ii)  $\tau(z) = \phi$  for all  $z \in [n] \setminus Y$ . For such a partial permutation  $\tau$ , we refer to the set  $X$  as its image, denoted  $\text{Image}(\tau)$ . A 0-1  $m \times n$  matrix  $\Pi$  is called a permutation matrix if every row and column has at most one entry that is 1 (all other entries are 0s). Note that there is an obvious correspondence between partial permutations and permutation matrices. In the rest of the paper we will use partial permutations that map a subset of jobs into a set of machines.

## 2 Randomized Algorithm for Minimizing Makespan

In this section, we give a randomized  $\Theta(\sqrt{\min\{m, n\}})$  approximation guarantee for minimizing makespan in the permutation flow shop problem. From the results of Potts et al. [21], it follows that this result is the best possible using the known lower bounds for this problem (namely, machine load & job length). Our algorithm is extremely simple: always output a permutation chosen uniformly at random. The rest of this section proves that this algorithm achieves a guarantee of  $2\sqrt{\min\{m, n\}}$ .

Given any instance of permutation flow shop, consider the  $m \times n$  matrix  $P$  of processing times. We first show how  $P$  can be decomposed into a collection of smaller matrices having certain properties.

**Lemma 1.** Let  $P \in \mathbb{N}^{m \times n}$ . Let  $h = \max\{l, L\}$ ,  $l = \max_{k=1}^h \sum_{i=1}^m p_{i,k}$ ,  $L = \max_{i=1}^m \sum_{j=1}^n p_{i,j}$

**Proof:** Define a bipartite multi-graph  $G$  corresponding to  $P$  as follows.  $G$  has vertex bipartition  $[m]$  and  $[n]$ . For every  $i \in [m]$  &  $j \in [n]$ ,  $G$  contains

$p_{i,j}$  parallel edges between  $i$  &  $j$ . Note that the maximum degree of  $G$  is exactly  $h = \max\{l, L\}$ . By the König edge-coloring theorem for bipartite graphs there is a valid coloring of the edges of  $G$  (no two adjacent edges receive the same color) with  $h$  colors. Let  $E_1, \dots, E_h$  denote the edges in each color class of such a valid coloring. For each  $1 \leq k \leq h$ , let  $\Pi_k$  denote the  $m \times n$  0-1 matrix that contains 1s in the positions corresponding to edges of  $E_k$ , and 0s everywhere else. Since we have a valid coloring, each  $E_k$  is a matching in  $G$ ; in other words, the matrices  $\{\Pi_k\}_{k=1}^h$  are all permutation matrices. Further, since each edge of  $G$  is assigned some color, we have  $P = \sum_{k=1}^h \Pi_k$ . ■

Recall that  $\mathcal{M}_{m,n}$  denotes the set of all monotone paths in an  $m \times n$  matrix. In this section,  $m$  and  $n$  are fixed; so we abbreviate  $\mathcal{M} = \mathcal{M}_{m,n}$ . For any permutation  $\sigma$  of the  $n$  jobs, monotone paths can be used to characterize the makespan of the schedule resulting from  $\sigma$  as follows:

$$C_{max}^\sigma = \max_{\alpha \in \mathcal{M}} \sum_{(i,q) \in \alpha} p_{i,\sigma(q)}$$

Consider any permutation matrix  $\Pi_k$  ( $k = 1, \dots, h$ ) in the decomposition of Lemma 1. Let the 1-entries of  $\Pi_k$  be in positions  $\{(x_1^k, y_1^k), \dots, (x_r^k, y_r^k)\}$ , where  $1 \leq x_1^k < \dots < x_r^k \leq m$  and  $y_1^k, \dots, y_r^k \in [n]$  are distinct elements. Denote  $X_k = \{x_1^k, \dots, x_r^k\}$  and  $Y_k = \{y_1^k, \dots, y_r^k\}$ ; clearly,  $|X_k| = |Y_k| = r \leq \min\{m, n\}$ . Define the map  $\tau_k : [n] \rightarrow X_k \cup \{\phi\}$  where  $\tau_k(y_g^k) = x_g^k$  (for all  $1 \leq g \leq r$ ) and  $\tau_k(z) = \phi$  for  $z \notin Y_k$ . Since each  $\Pi_k$  is a permutation matrix, it follows that the  $\tau_k$  is a partial permutation for  $k = 1, \dots, h$ .

Finally, for any sequence  $S$  of elements from  $[m] \cup \phi$ , define  $I(S)$  to be the length of the longest increasing subsequence of numbers in  $S$  (ignoring all occurrences of the null element  $\phi$ ).

**Lemma 2.**  $\sum_{(i,q) \in \alpha} p_{i,\sigma(q)} \leq \sum_{k=1}^h I(\tau_k \circ \sigma[n])$   $\alpha \in \mathcal{M}$

$$\sum_{(i,q) \in \alpha} p_{i,\sigma(q)} \leq \sum_{k=1}^h I(\tau_k \circ \sigma[n])$$

**Proof:** Clearly we have:

$$\sum_{(i,q) \in \alpha} p_{i,\sigma(q)} = \sum_{(i,q) \in \alpha} \left[ \sum_{k=1}^h \Pi_k(i, \sigma(q)) \right] = \sum_{k=1}^h \sum_{(i,q) \in \alpha} \Pi_k(i, \sigma(q))$$

Now consider a particular permutation matrix  $\Pi_k$  (for  $k = 1, \dots, h$ ) and the sum  $\sum_{(i,q) \in \alpha} \Pi_k(i, \sigma(q))$ .

Let  $S_k = \{(i, q) \in \alpha \mid \Pi_k(i, \sigma(q)) = 1\}$ ; then the sum  $\sum_{(i,q) \in \alpha} \Pi_k(i, \sigma(q)) = |S_k|$ . Since  $\Pi_k$  has at most one non-zero entry in each row and column (given by the partial permutation  $\tau_k$ ) and  $\alpha$  is a monotone path, we obtain that  $S_k = \{(i_1, q_1), \dots, (i_t, q_t)\}$  (where  $t = |S_k|$ ), with the following properties:

1.  $1 \leq i_1 < \dots < i_t \leq m$  and  $\{i_1, \dots, i_t\} \subseteq X_k$ .
2.  $1 \leq q_1 < \dots < q_t \leq n$ .
3.  $\tau_k(\sigma(q_g)) = i_g$  for all  $1 \leq g \leq t$ .

From the above, we have that  $i_1 < i_2 < \dots < i_t$  is an increasing subsequence of length  $t$  in the sequence  $\tau_k \circ \sigma[n] = \langle \tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(n) \rangle$ ; namely given by the positions  $q_1 < q_2 < \dots < q_t$ . Thus the longest increasing subsequence in  $\tau_k \circ \sigma[n]$  has length at least  $|S_k|$ . In other words,  $\sum_{(i,q) \in \alpha} \Pi_k(i, \sigma(q)) \leq I(\tau_k \circ \sigma[n])$ . Summing this expression over all permutation matrices  $\Pi_k$  for  $k = 1, \dots, h$ , we obtain the statement of the Lemma. ■

Note that the right hand side in the inequality in Lemma 2 does not depend on the monotone path  $\alpha$ ; hence we obtain that  $C_{max}^\sigma = \max_{\alpha \in \mathcal{M}} \sum_{(i,q) \in \alpha} p_{i,\sigma(q)} \leq \max_{\alpha \in \mathcal{M}} \sum_{k=1}^h I(\tau_k \circ \sigma[n]) = \sum_{k=1}^h I(\tau_k \circ \sigma[n])$ . We will also need the following:

**Lemma 3 (Logan & Shepp [16]; Vershik & Kerov [31]).**  $E_\sigma \left[ \sum_{(i,q) \in \alpha} p_{i,\sigma(q)} \right] \leq (2 + o(1)) \sqrt{r}$

We are now ready for the main theorem of this section.

**Theorem 1.**  $E_\sigma [C_{max}^\sigma] \leq (2 + o(1)) h \cdot \sqrt{\min\{m, n\}}$

**Proof:** From the linearity of expectation, Lemma 2 and the comment following it, it suffices to bound  $E_\sigma [I(\tau_k \circ \sigma[n])]$  for each  $1 \leq k \leq h$ . Fix a  $1 \leq k \leq h$ : since  $\sigma$  is chosen uniformly at random over all permutations, the jobs from  $Y_k$  are ordered uniformly at random. Thus  $\tau_k \circ \sigma[n]$  is a uniformly random ordering of the elements  $X_k$  (ignoring occurrences of  $\phi$ ). Applying Lemma 3, we immediately obtain the following which proves the theorem.

$$E_\sigma [I(\tau_k \circ \sigma[n])] \leq (2 + o(1)) \sqrt{|X_k|} \leq (2 + o(1)) \sqrt{\min\{m, n\}}.$$

Thus we have a very simple randomized  $\Theta(\sqrt{\min\{m, n\}})$ -approximation algorithm for the permutation flow shop problem, based on the trivial lower bound. Potts et al. [21] gave a family of examples where the optimal permutation schedule has length at least  $\frac{1}{\sqrt{2}} \sqrt{\min\{m, n\}}$  times the lower bound. Hence our result is the best possible guarantee (within a factor of  $2\sqrt{2}$ ) using these lower bound. We note that Theorem 1 also implies that for any instance of flow shop scheduling, there is a permutation schedule of length at most  $2\sqrt{\min\{m, n\}}$  times the length of an optimal schedule; hence this resolves positively the open question in Potts et al. [21] regarding the gap between permutation & non-permutation schedules.

**Tight Example.** The following simple example shows that the performance guarantee of this randomized algorithm is tight. There are  $n$  jobs and  $m = 2n$

machines. Each job  $j$  (for  $1 \leq j \leq n$ ) has processing time 1 on machines  $j$  and  $n + j$ , and 0 elsewhere. The optimal permutation of jobs is  $n, n - 1, \dots, 1$  which results in a makespan of 2. However, it follows from Lemma 3 that a random permutation has expected makespan at least  $2\sqrt{n}$ .

### 3 A Deterministic Algorithm

We apply the technique of [24] due to Raghavan to derandomize the algorithm of the previous section, and obtain a deterministic  $\Theta(\sqrt{\min\{m, n\}})$ -approximation guarantee. We first apply the decomposition of Lemma 1 to obtain  $h$  permutation-matrices  $\Pi_1, \dots, \Pi_h$  corresponding to  $P$ . By assigning weights  $w_1, \dots, w_h \in \mathbb{N}$  to each of these permutations, we can ensure that  $P = \sum_{k=1}^h w_k \cdot \Pi_k$  and  $h \leq mn$ ; here  $\sum_{k=1}^h w_k$  is the trivial lower-bound for the flowshop instance. This computation can be done easily in polynomial time by iteratively using any bipartite matching algorithm. There are many more efficient algorithms for computing an edge-coloring in bipartite multigraphs (See the table in Section 20.9b [27] for running times and references for various edge-coloring algorithms). Further Lemma 2 implies that for any permutation  $\sigma : [n] \rightarrow [n]$  of the jobs, the resulting makespan  $C_{max}^\sigma \leq C^*(\sigma) \doteq \sum_{k=1}^h w_k \cdot I(\tau_k \circ \sigma)$ , where  $\tau_k$ s are the partial permutations corresponding to the permutation-matrices  $\Pi_k$ s. From the previous section, we have that  $E_\sigma[C^*(\sigma)] \leq 2\sqrt{\min\{m, n\}} \cdot \sum_{k=1}^h w_k$ . In this section, we give a deterministic algorithm that obtains a permutation  $\sigma$  satisfying  $C^*(\sigma) \leq 3\sqrt{\min\{m, n\}} \cdot \sum_{k=1}^h w_k$ .

In particular, we show that given any collection of  $h$  partial permutations  $\tau_1, \dots, \tau_h : [n] \rightarrow [m] \cup \{\phi\}$ , each having a non-empty value on at most  $r$  elements, and associated weights  $\{w_k\}_{k=1}^h$ , there is a polynomial time deterministic algorithm that computes a single permutation  $\sigma : [n] \rightarrow [n]$  satisfying  $C^*(\sigma) = \sum_{k=1}^h w_k \cdot I(\tau_k \circ \sigma) \leq 3\sqrt{r} \cdot \sum_{k=1}^h w_k$ . This immediately implies the desired deterministic approximation guarantee for the permutation flow shop problem since each partial permutation has an image of size at most  $r \leq \min\{m, n\}$ . In the following, we refer to a permutation that is chosen uniformly at random as u.a.r. permutation.

The algorithm first computes the partial permutations  $\tau_k$  and weights  $w_k$  for  $k = 1, \dots, h$ , and then builds the solution  $\sigma$  incrementally. In each step  $i = 1, \dots, n$  we suitably fix the value of  $\sigma(i)$  that results in a prefix of the permutation  $\langle \sigma(1), \dots, \sigma(i) \rangle$ , i.e. we fix jobs located in the first  $i$  positions. The choices for  $\sigma(i)$  in each step  $i$  are made in such a way that finally,  $C^*(\sigma) \leq 3\sqrt{r} \cdot \sum_{k=1}^h w_k$ . Define the following quantities for any partial permutation  $\tau_k$  ( $1 \leq k \leq h$ ), step  $0 \leq i \leq n$ , and elements  $a_1, \dots, a_i \in \text{Image}(\tau_k) \cup \{\phi\}$ :

- expected value of the longest increasing subsequence
- $E_i^k(a_1, \dots, a_i) \doteq \text{in } \langle a_1, \dots, a_i, \tau \rangle$ , where  $\tau$  is a permutation on  $\text{Image}(\tau_k) \setminus \{a_1, \dots, a_i\}$  picked u.a.r.
- $U_i^k(a_1, \dots, a_i) \doteq$  an efficiently computable upper bound on  $E_i^k(a_1, \dots, a_i)$  (exact definition later).



In the above definitions, the elements  $a_1, \dots, a_i$  represent the values  $\tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(i)$  respectively, obtained from the first  $i$  positions of permutation  $\sigma$ , that have been fixed thus far. We also define the expected value of function  $C^*(\sigma)$  in step  $i$  as functions of the first  $i$  positions of permutation  $\sigma$  (that have been fixed):

$E_i(\sigma(1), \dots, \sigma(i))$ : expected value  $E_{\sigma(i+1)\dots\sigma(n)}[C^*(\sigma)]$ , with  $\langle \sigma(i+1) \dots \sigma(n) \rangle$  being a u.a.r. permutation on  $[n] \setminus \{\sigma(1), \dots, \sigma(i)\}$

Note that for any  $1 \leq k \leq h$ , since  $\langle \sigma(i+1), \dots, \sigma(n) \rangle$  is u.a.r. permutation on  $[n] \setminus \{\sigma(1), \dots, \sigma(i)\}$ , we obtain that  $\langle \tau_k \circ \sigma(i+1), \dots, \tau_k \circ \sigma(n) \rangle$  is u.a.r. permutation on  $\text{Image}(\tau_k) \setminus \{\tau_k \circ \sigma(j) : 1 \leq j \leq i\}$ . Thus we can rewrite  $E_i$  as:

$$E_i(\sigma(1), \dots, \sigma(i)) \doteq \sum_{k=1}^h w_k \cdot E_i^k(\tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(i))$$

We also define the efficiently computable upper bound on  $E_i$  as:

$$U_i(\sigma(1), \dots, \sigma(i)) \doteq \sum_{k=1}^h w_k \cdot U_i^k(\tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(i))$$

The precise definition of the upper bound functions  $U_i^k$  for  $k = 1, \dots, h$  and  $i = 0, \dots, n$  appears in the next subsection. In Lemmas 4 and 5, we prove some important properties of the functions  $U_i$ , which allow us to derandomize the algorithm of the previous section to obtain Theorem 2.

### 3.1 Properties of the Pessimistic Estimator

Recall the definition of  $E_i^k(a_1, \dots, a_i)$ ; we now construct an upper bound  $U_i^k(a_1, \dots, a_i)$  for this expected value. Fix a parameter  $t = 3\sqrt{r}$ , where  $r = \max_{k=1}^h |\text{Image}(\tau_k)|$  is an upper bound on the length of each partial permutation. Define  $S_i^k(a_1, \dots, a_i)$  to be the length of the longest increasing subsequences in  $\langle a_1, \dots, a_i, \tau \rangle$ , when  $\tau$  is u.a.r. permutation on  $\text{Image}(\tau_k) \setminus \{a_1, \dots, a_i\}$ . We can now upper bound  $E_i^k(a_1, \dots, a_i)$ , the expected length of the longest increasing subsequence in  $\langle a_1, \dots, a_i, \tau \rangle$ , as follows:

$$\begin{aligned} E_i^k(a_1, \dots, a_i) &\leq t \cdot \Pr_{\tau}[\langle a_1, \dots, a_i, \tau \rangle \text{ has no } t\text{-length increasing subsequence}] \\ &\quad + r \cdot \Pr_{\tau}[\langle a_1, \dots, a_i, \tau \rangle \text{ contains a } t\text{-length increasing subsequence}] \\ &\leq t + r \cdot \Pr_{\tau}[\langle a_1, \dots, a_i, \tau \rangle \text{ contains a } t\text{-length increasing subsequence}] \\ &\leq t + r \cdot S_i^k(a_1, \dots, a_i) \end{aligned}$$

Define the upper bound  $U_i^k$  on the expected value  $E_i^k$  as:

$$U_i^k(a_1, \dots, a_i) \doteq t + r \cdot S_i^k(a_1, \dots, a_i) \quad \forall 1 \leq k \leq h$$

Let  $N_i^k = \text{Image}(\tau_k) \setminus \{a_1, \dots, a_i\} \subseteq [m]$ ; and for any set  $T$ , let  $\mathcal{P}(T)$  denote the set of all permutations of the elements of  $T$ . We first show that each  $U_i^k$  can be efficiently computed, which implies the same for the functions  $\{U_i\}_{i=0}^n$ .



**Lemma 4.**  $1 \leq k \leq h$   $i \in \{0, \dots, n\}$   $a_1, \dots, a_i \in \text{Image}(\tau_k) \cup \{\phi\}$   
 $U_i^k(a_1, \dots, a_i)$

**Proof:** Fix any values of  $1 \leq k \leq h$ ,  $0 \leq i \leq n$  and  $a_1, \dots, a_i \in \text{Image}(\tau_k) \cup \{\phi\}$ . Clearly it suffices to show that  $S_i^k(a_1, \dots, a_i)$  can be computed in polynomial time. We say that a  $t$ -length increasing subsequence  $s$  is  $\dots$  if there is some permutation  $\tau \in \mathcal{P}(N_i^k)$  such that  $s$  is a subsequence in  $\langle a_1, \dots, a_i, \tau \rangle$ . Let  $\mathcal{I}$  denote the set of all such feasible  $t$ -length increasing subsequences. Then we can partition  $\mathcal{I}$  as  $(\sqcup \{\mathcal{I}_{j,q} \mid 1 \leq j \leq i \ \& \ 1 \leq q \leq t\}) \sqcup \mathcal{I}_{0,0}$  where:

$$\mathcal{I}_{0,0} = \{\tau^0 \mid \tau^0 \text{ is a } t\text{-length increasing sequence of numbers from } N_i^k\}$$

$$\mathcal{I}_{j,q} = \left\{ \langle \tau', \tau'' \rangle \mid \begin{array}{l} \tau' \text{ is a } q \text{ length increasing subsequence in } \langle a_1, \dots, a_j \rangle \\ \text{ending at } a_j \neq \phi, \text{ and } \tau'' \text{ is a } t - q \text{ length increasing} \\ \text{sequence of numbers from } \{e \in N_i^k : e > a_j\} \end{array} \right\}$$

Note that given any  $j \in \{1, \dots, i\}$  and  $q \in \{1, \dots, t\}$ , one can compute in polynomial time, the number of  $q$ -length increasing subsequences in  $\langle a_1, \dots, a_j \rangle$  that end at  $a_j$ ; we denote this quantity by  $\#I(j, q)$ . The computation of  $\#I(j, q)$  is based on a dynamic program using the following recurrence:

$$\#I(j, q) = \begin{cases} \sum \{\#I(j', q - 1) \mid 1 \leq j' < j, a_{j'} < a_j\} & a_j \neq \phi, q \geq 2 \\ 1 & a_j \neq \phi, q = 1 \\ 0 & a_j = \phi \end{cases}$$

For ease of notation in the following, let  $\#I(0, 0) = 1$ . For every  $1 \leq j \leq i$ , denote the set  $\{e \in N_i^k : e > a_j\}$  by  $L_j$ , and also let  $L_0 = N_i^k$ . Note that, for each part  $\mathcal{I}_{j,q}$  (in the partition of  $\mathcal{I}$ ), its size  $|\mathcal{I}_{j,q}| = \#I(j, q) \cdot \binom{|L_j|}{t-q}$  (the first term corresponds to a  $q$  length increasing subsequence of  $\langle a_1, \dots, a_j \rangle$ , and the second term corresponds to a  $t - q$  length increasing sequence from  $L_j$ ). When  $\tau \in \mathcal{P}(N_i^k)$  is picked u.a.r., the induced permutation on each set  $L_j$  (for  $0 \leq j \leq i$ ) is also u.a.r. Hence for each part  $\mathcal{I}_{j,q}$ , the probability that any particular subsequence  $s \in \mathcal{I}_{j,q}$  appears in  $\langle a_1, \dots, a_i, \tau \rangle$  is exactly  $1/(t - q)!$ . (the last  $t - q$  entries of  $s$  come from the random permutation  $\tau$ ). So we have:

$$\begin{aligned} E_\tau [|\{s \in \mathcal{I}_{j,q} : s \text{ is subsequence of } \langle a_1, \dots, a_i, \tau \rangle\}|] &= \sum_{s \in \mathcal{I}_{j,q}} Pr_\tau [s \text{ is subsequence of } \langle a_1, \dots, a_i, \tau \rangle] \\ &= |\mathcal{I}_{j,q}| \cdot \frac{1}{(t-q)!} \\ &= \#I(j, q) \cdot \binom{|L_j|}{t-q} \cdot \frac{1}{(t-q)!} \end{aligned}$$

Thus, we can write  $S_i^k(a_1, \dots, a_i)$  as

$$\begin{aligned} E_{\tau \in \mathcal{P}(N_i^k)} [|\{s \in \mathcal{I} : s \text{ is subsequence of } \langle a_1, \dots, a_i, \tau \rangle\}|] &= \sum_{j,q} E_{\tau \in \mathcal{P}(N_i^k)} [|\{s \in \mathcal{I}_{j,q} : s \text{ is subsequence of } \langle a_1, \dots, a_i, \tau \rangle\}|] \\ &= \sum_{j,q} \#I(j, q) \cdot \binom{|L_j|}{t-q} \cdot \frac{1}{(t-q)!} \end{aligned}$$

The lemma follows. ■

**Lemma 5.**  $0 \leq i \leq n$ ,  $\sigma(1), \dots, \sigma(i) \in [n]$

$$\min_{\sigma(i+1) \in [n] \setminus \{\sigma(1), \dots, \sigma(i)\}} U_{i+1}(\sigma(1), \dots, \sigma(i), \sigma(i+1)) \leq U_i(\sigma(1), \dots, \sigma(i))$$

**Proof:** Fix any  $i$  and a prefix  $\sigma(1), \dots, \sigma(i)$  of a permutation  $\sigma$ , and let  $M = [n] \setminus \{\sigma(1), \dots, \sigma(i)\}$ . We first prove the following for an arbitrary  $1 \leq k \leq h$ :

$$S_i^k(\tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(i)) = \frac{1}{n-i} \sum_{x \in M} S_{i+1}^k(\tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(i), \tau_k(x)) \tag{1}$$

For ease of notation, let  $a_j^k = \tau_k \circ \sigma(j)$  for all  $1 \leq j \leq i$ . Let  $N_i^k = \text{Image}(\tau_k) \setminus \{a_1^k, \dots, a_i^k\} \subseteq [m]$  denote the remaining elements of  $\text{Image}(\tau_k)$ , and  $n_i^k = |N_i^k|$ . Recall that,

$$S_i^k(a_1^k, \dots, a_i^k) = E_\tau[\text{number of } t\text{-length increasing subsequences in } \langle a_1^k \dots a_i^k, \tau \rangle]$$

where  $\tau \in \mathcal{P}(N_i^k)$  is picked u.a.r. So multiplying both sides of (1) by  $n_i^k! = |\mathcal{P}(N_i^k)|$ , we can rewrite its left hand side as:

$$LHS' = n_i^k! \times S_i^k(a_1^k, \dots, a_i^k) = \sum_{\tau \in \mathcal{P}(N_i^k)} \#I_t(a_1^k, \dots, a_i^k, \tau) \tag{2}$$

Above, for any sequence  $s$ ,  $\#I_t(s)$  denotes the number of  $t$ -length increasing subsequences in  $s$ . To compute the right hand side of (1), we split the summation into  $M^{(k)} = \{x \in M \mid \tau_k(x) \neq \phi\}$  and  $M \setminus M^{(k)} = \{x \in M \mid \tau_k(x) = \phi\}$ . Note that  $|M| = n - i$  and  $|M^{(k)}| = n_i^k$ . For any  $x \in M \setminus M^{(k)}$ , it is easy to see that  $S_{i+1}^k(a_1^k, \dots, a_i^k, \tau_k(x)) = S_i^k(a_1^k, \dots, a_i^k)$ . Now the right hand side of (1) (scaled by  $n_i^k!$ ) can be written as:

$$\begin{aligned} & n_i^k! \times \frac{n-i-n_i^k}{n-i} S_i^k(a_1^k, \dots, a_i^k) + n_i^k! \times \frac{1}{n-i} \sum_{x \in M^{(k)}} S_{i+1}^k(a_1^k, \dots, a_i^k, \tau_k(x)) \\ &= (1 - \frac{n_i^k}{n-i}) LHS' + n_i^k! \times \frac{1}{n-i} \sum_{x \in M^{(k)}} S_{i+1}^k(a_1^k, \dots, a_i^k, \tau_k(x)) \end{aligned}$$

Thus in order to prove (1), it suffices to show:

$$LHS' = (n_i^k - 1)! \times \sum_{x \in M^{(k)}} S_{i+1}^k(a_1^k, \dots, a_i^k, \tau_k(x)) \tag{3}$$

Note that  $\tau_k$  induces a bijection between  $M^{(k)}$  and  $N_i^k$ :  $|M^{(k)}| = |N_i^k|$  and  $\tau_k(M^{(k)}) = N_i^k$ . Thus we can rewrite the right hand side in (3) as:

$$(n_i^k - 1)! \sum_{y \in N_i^k} S_{i+1}^k(a_1^k, \dots, a_i^k, y) = \sum_{y \in N_i^k} \sum_{\tau' \in \mathcal{P}(N_i^k \setminus y)} \#I_t(a_1^k, \dots, a_i^k, y, \tau')$$

To prove (3), using the expression for  $LHS'$  from (2), it suffices to show that:

$$\sum_{\tau \in \mathcal{P}(N_i^k)} \#I_t(a_1^k, \dots, a_i^k, \tau) = \sum_{y \in N_i^k} \sum_{\tau' \in \mathcal{P}(N_i^k \setminus y)} \#I_t(a_1^k, \dots, a_i^k, y, \tau')$$

Now observe that  $\mathcal{P}(N_i^k) = \sqcup_{y \in N_i^k} \{\langle y, \tau' \rangle \mid \tau' \in \mathcal{P}(N_i^k \setminus y)\}$ . Thus the summations in the two expressions above run over exactly the same set of sequences, and this implies equality (3) which in turn gives equation (4). We are now ready to complete the proof of the lemma.

$$\begin{aligned} \min_{\sigma(i+1) \in M} U_{i+1}(\sigma(1), \dots, \sigma(i), \sigma(i+1)) &\leq \frac{1}{n-i} \sum_{x \in M} U_{i+1}(\sigma(1), \dots, \sigma(i), x) \\ &= \frac{1}{n-i} \sum_{x \in M} \sum_{k=1}^h w_k [t + r \cdot S_{i+1}^k(\tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(i), \tau_k(x))] \\ &= \frac{|M|}{n-i} \cdot t \sum_{k=1}^h w_k + \frac{1}{n-i} \sum_{x \in M} r \cdot \sum_{k=1}^h w_k \cdot S_{i+1}^k(\tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(i), \tau_k(x)) \\ &= t \sum_{k=1}^h w_k + r \cdot \sum_{k=1}^h w_k \cdot \frac{1}{n-i} \sum_{x \in M} S_{i+1}^k(\tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(i), \tau_k(x)) \\ &= t \sum_{k=1}^h w_k + r \cdot \sum_{k=1}^h w_k \cdot S_i^k(\tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(i)) \quad (\text{Using equation (4)}) \\ &= \sum_{k=1}^h w_k \cdot U_i^k(\tau_k \circ \sigma(1), \dots, \tau_k \circ \sigma(i)) \\ &= U_i(\sigma(1), \dots, \sigma(i)) \end{aligned}$$

Thus we have the lemma. ■

### 3.2 Applying the Pessimistic Estimators

We now use the upper-bound functions  $U_i$  for  $i = 1, \dots, n$  described in the previous subsection to obtain a deterministic approximation algorithm for the permutation flow shop problem. This algorithm follows the general framework of the method of pessimistic estimators.

**Theorem 2.** *Let  $m, n$  be the number of machines and jobs respectively. Then there exists a deterministic algorithm that approximates the makespan of a permutation flow shop instance with  $m$  machines and  $n$  jobs within a factor of  $3\sqrt{\min\{m, n\}}$ .*

**Proof:** We now describe our final deterministic algorithm:

1. Decompose the matrix  $P$  of processing times according to Lemma 1, to obtain  $h \leq mn$  permutation-matrices with corresponding weights  $\{\Pi_k, w_k\}_{k=1}^h$ , such that  $P = \sum_{k=1}^h w_k \cdot \Pi_k$  and  $\sum_{k=1}^h w_k$  equals the trivial lower-bound for the permutation flow shop instance.

2. For each  $1 \leq k \leq h$ ,  $\tau_k$  denotes the partial permutation corresponding to permutation-matrix  $\Pi_k$ .
3. For each  $i = 1, \dots, n$ : set  $\sigma(i) \leftarrow x$  for the value  $x \in [n] \setminus \{\sigma(1), \dots, \sigma(i-1)\}$  that minimizes the function value  $U_i(\sigma(1), \dots, \sigma(i-1), x)$ .

As mentioned earlier, the decomposition in step 1 can be carried out in polynomial time using an edge-coloring algorithm. In step 3, the algorithm uses the efficiently computable functions  $\{U_i\}_{i=0}^n$  (see Lemma 4) to fix the solution  $\sigma$  step by step. Hence the above algorithm runs in polynomial time. The rest of this proof shows that it achieves the desired approximation guarantee.

We claim that for each  $i \in \{0, \dots, n\}$ ,  $U_i(\sigma(1), \dots, \sigma(i)) \leq W \cdot (t + 2)$  where  $W = \sum_{k=1}^h w_k$  is the trivial lower-bound (recall that  $t = 3\sqrt{r} \leq 3\sqrt{\min\{m, n\}}$ ). Assuming that the base case (i.e.  $i = 0$ ) for this claim holds, using Lemma 5 and induction, we obtain that the claim is true for all values of  $i \geq 1$ . It remains to prove the claim for  $i = 0$ : here  $U_0$  takes no arguments and is a fixed value  $U_0 = tW + r \sum_{k=1}^h w_k \cdot S_0^k$ . From the definition of the  $S_i^k$ s, we have that each  $S_0^k$  is the expected number of  $t$ -length increasing subsequences in a u.a.r. permutation of  $\text{Image}(\tau_k)$ . Since  $\text{Image}(\tau_k)$  has at most  $r$  elements, using linearity of expectation, it follows that  $S_0^k \leq \binom{r}{t} \cdot \frac{1}{t!}$  for every  $k = 1, \dots, h$ . We have,

$$\begin{aligned}
 U_0 &\leq tW + rW \binom{r}{t} \frac{1}{t!} = tW + rW \frac{r!}{(r-t)!t!} \frac{1}{t!} \leq tW + rW \frac{r^t}{(t!)^2} \\
 &\leq tW + rW \left(\frac{re^2}{t^2}\right)^t = tW + rW \left(\frac{e^2}{9}\right)^t = tW + rW \left(\frac{e}{3}\right)^{6\sqrt{r}} \leq W \cdot (t + 2)
 \end{aligned}$$

Now observe that after the last step,  $E_n(\sigma(1), \dots, \sigma(n))$  is exactly the value  $C^*(\sigma)$  (at this point all positions have been fixed, so there is no randomness left in the expected value  $E_n$ ). Since the function  $U_n$  upper bounds  $E_n$ , we have  $C^*(\sigma) = E_n(\sigma(1), \dots, \sigma(n)) \leq U_n(\sigma(1), \dots, \sigma(n)) \leq W \cdot (t + 2)$ . Now the theorem follows from the fact that  $W$  equals the trivial lower-bound for the permutation flow shop instance and  $r \leq \min\{m, n\}$ . ■

### 4 Weighted Sum of Completion Times

In this section, we consider the permutation flow shop problem with the objective being the weighted sum of completion times. We show that our algorithm for the makespan objective can be used within an LP-based approach to obtain an  $O(\sqrt{\min\{m, n\}})$  approximation algorithm for weighted completion time. This approach is similar to that used in Queyranne and Sviridenko [23] (and many other papers on scheduling with the weighted completion time objective [11, 10]), where the authors considered a class of job shop problems (these do not have the permutation constraint). We consider the following linear relaxation for the permutation flow shop problem with weighted completion time as objective. In fact this LP is a relaxation for even the usual flow shop problem (without the permutation constraint), and is a special case of the LP studied in [23].

$$\min \sum_{j=1}^n w_j \cdot C_j, \quad (4)$$

$$z_{1,j} \geq p_{1,j}, \quad \forall 1 \leq j \leq n \quad (5)$$

$$z_{i,j} \geq z_{i-1,j} + p_{i,j}, \quad \forall 2 \leq i \leq m, 1 \leq j \leq n \quad (6)$$

$$\sum_{j \in A} p_{i,j} \cdot z_{i,j} \geq \frac{1}{2} \left( \sum_{j \in A} p_{i,j} \right)^2 + \frac{1}{2} \sum_{j \in A} p_{i,j}^2, \quad \forall A \subseteq [n], 1 \leq i \leq m, \quad (7)$$

$$C_j = z_{m,j}, \quad \forall 1 \leq j \leq n \quad (8)$$

$$z_{i,j} \geq 0, \quad \forall 1 \leq i \leq m, 1 \leq j \leq n \quad (9)$$

Using this LP, the standard geometric partitioning technique and our algorithm for the problem with the makespan objective we were able to prove.

**Theorem 3.**  $O(\sqrt{\min\{m, n\}})$

## References

1. Chakrabarti, S., Phillips, C., Schulz, A., Shmoys, D., Stein, C., Wein, J.: Improved Scheduling Algorithms for Minsum Criteria. In: Meyer auf der Heide, F., Monien, B. (eds.) ICALP 1996. LNCS, vol. 1099, pp. 646–657. Springer, Heidelberg (1996)
2. Chazelle, B.: The discrepancy method. Randomness and complexity. Cambridge University Press, Cambridge (2000)
3. Chen, B., Potts, C., Woeginger, G.: A review of machine scheduling: complexity, algorithms and approximability. In: Du, D.-Z., Pardalos, P.M. (eds.) Handbook of combinatorial optimization, vol. 3, pp. 21–169. Kluwer Academic Publishers, Boston (1998)
4. Conway, R., Maxwell, W., Miller, L.: Theory of scheduling. Addison-Wesley Publishing Co., Reading, Mass, London, Don Mills, Ont. (1967)
5. Czumaj, A., Scheideler, C.: A New Algorithmic Approach to the General Lovasz Local Lemma with Applications to Scheduling and Satisfiability Problems. In: Proc. 32 ACM Symposium on Theory of Computing (STOC) (2000)
6. Feige, U., Scheideler, C.: Improved bounds for acyclic job shop scheduling. In: STOC 1998. Proceedings of the 30th Annual ACM Symposium on Theory of Computing, pp. 624–633. ACM Press, New York (1998)
7. Fishkin, A., Jansen, K., Mastrolilli, M.: On minimizing average weighted completion time: a PTAS for the job shop problem with release dates. In: Ibaraki, T., Katoh, N., Ono, H. (eds.) ISAAC 2003. LNCS, vol. 2906, pp. 319–328. Springer, Heidelberg (2003)
8. Framinan, J., Gupta, J., Leisten, R.: A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. Journal of the Operational Research Society 55, 1243–1255 (2004)
9. Frieze, A.: On the length of the longest monotone subsequence of a random permutation. The Annals of Applied Probability 1(2), 301–305 (1991)
10. Hall, L.A., Shmoys, D.B., Wein, J.: Scheduling to Minimize Average Completion Time: Off-line and On-line Algorithms. In: Proceedings of the 7th Symposium on Discrete Algorithms, pp. 142–151 (1996)

11. Hall, L.A., Schulz, A.S., Shmoys, D.B., Wein, J.: Scheduling to Minimize Average Completion Time: Off-Line and On-Line Approximation Algorithms. *Mathematics of Operations Research* 22, 513–544 (1997)
12. Jansen, K., Solis-Oba, R., Sviridenko, M.: Makespan Minimization in Job Shops: a Linear Time Approximation Scheme. *SIAM Journal of Discrete Mathematics* 16, 288–300 (2003)
13. Johnson, S.: Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly* 1, 61–68 (1954)
14. Hofri, M.: *Probabilistic Analysis of Algorithms: On Computing Methodologies for Computing Algorithms Performance Evaluation*. Springer, Heidelberg (1987)
15. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and scheduling: Algorithms and complexity. In: *Handbook in Operations Research and Management Science*, vol. 4, pp. 445–522. North-Holland, Amsterdam (1993)
16. Logan, B.F., Shepp, L.A.: A Variational Problem for Random Young Tableaux. *Advances in Mathematics* 26, 206–222 (1977)
17. Nawaz, M., Ensore Jr., E., Ham, I.: A heuristic algorithm for the m-machine n-job flow-shop sequencing problem. *OMEGA International J. Management Sci.* 11, 91–95 (1983)
18. Nowicki, E., Smutnicki, C.: New results in the worst-case analysis for flow-shop scheduling. *Discrete Appl. Math.* 46, 21–41 (1993)
19. Nowicki, E., Smutnicki, C.: Worst-case analysis of an approximation algorithm for flow-shop scheduling. *Oper. Res. Lett.* 8, 171–177 (1989)
20. Nowicki, E., Smutnicki, C.: Worst-case analysis of Dannenbring's algorithm for flow-shop scheduling. *Oper. Res. Lett.* 10, 473–480 (1991)
21. Potts, C., Shmoys, D., Williamson, D.: Permutation vs. nonpermutation flow shop schedules. *Operations Research Letters* 10, 281–284 (1991)
22. Queyranne, M.: Structure of a simple scheduling polyhedron. *Math. Programming Ser. A* 58(2), 263–285 (1993)
23. Queyranne, M., Sviridenko, M.: Approximation Algorithms for Shop Scheduling Problems with Minsum Objective. *Journal of Scheduling* 5, 287–305 (2002)
24. Raghavan, P.: Probabilistic construction of deterministic algorithms: approximating packing integer programs. *J. Comput. System Sci.* 37, 130–143 (1988)
25. Röck, H., Schmidt, G.: Machine aggregation heuristics in shop-scheduling. *Methods of Operations Research* 45, 303–314 (1983)
26. Sevast'janov, S.: On some geometric methods in scheduling theory: a survey. *Discrete Applied Mathematics* 55, 59–82 (1994)
27. Schrijver, A.: Combinatorial optimization. Polyhedra and efficiency. In: *Algorithms and Combinatorics*, vol. B 24. Springer, Berlin (2003)
28. Shmoys, D., Stein, C., Wein, J.: Improved Approximation Algorithms for Shop Scheduling Problems. *SIAM Journal on Computing* 23(3), 617–632 (1994)
29. Sviridenko, M.: A Note on Permutation Flow Shop Problem. *Annals of Operations Research* 129, 247–252 (2004)
30. Smutnicki, C.: Some results of the worst-case analysis for flow shop scheduling. *European Journal of Operational Research* 109, 66–87 (1998)
31. Vershik, A.M., Kerov, S.V.: Asymptotics of the Plancherel measure of the symmetric group and the limit form of Young tableaux. *Dokl. Akad. Nauk SSSR* 233, 1024–1027 (1977)

# The Stochastic Machine Replenishment Problem

Kamesh Munagala\* and Peng Shi\*\*

Department of Computer Science, Duke University, Durham NC 27708-0129  
kamesh@cs.duke.edu, peng.shi@duke.edu

**Abstract.** We study the stochastic machine replenishment problem, which is a canonical special case of closed multiclass queuing systems in Markov decision theory. The problem models the scheduling of processor repairs in a multiprocessor system in which one repair can be made at a time and the goal is to maximize system utilization. We analyze the performance of a natural greedy index policy for this problem. We first show that this policy is a 2 approximation by exploring linear queuing structure in the index policy. We then try to exploit more complex queuing structures, but this necessitates solving an infinite-size, non-linear, non-convex, and non-separable function-maximization program. We develop a general technique to solve such programs to arbitrary degree of accuracy, which involves solving a discretized program on the computer and rigorously bounding the error. This proves that the index policy is in fact a 1.51 approximation.

The main non-trivial ingredients of the proof are two folds: finding a way to analyze the complex queuing structure of the index policy, and bounding the error in discretization when numerically solving the non-linear function-maximization. We believe that this framework is general enough to be useful in the analysis of index policies in related problems. As far as we are aware, this is one of the first non-trivial approximation analysis of an index policy a multi-class queuing problem, as well as one of the first non-trivial example of an approximation ratio that is rigorously proven by numerical optimization via a computer.

## 1 Introduction

We study the performance of greedy priority schemes (or index policies) for the machine replenishment problem, which is a canonical special case of the well-studied but notoriously intractable closed multi-class queuing systems [3,11,17] in decision theory. This problem models how to maximize output in a multi-machine system in which machines can break down and there's a maintenance constraint. The problem is formally stated below.

There are  $n$  machines, labelled  $1, 2, \dots, n$ . Each machine can be either "active" or "broken", and all machines are active at the start of the process. When machine  $i$  is active, it yields reward at rate  $r_i$ ; when it's broken, it yields

---

\* Supported by NSF grant CNS-0540347.

\*\* This research was supported by the Duke University Work-Study Program.

nothing. If machine  $i$  is currently active, it transitions to the broken state according to a memoryless random process, independent from the other machines, at rate  $p_i$ . (The time it takes for machine  $i$  to break down is an independent random variable distributed as  $\text{Exponential}(p_i)$ ).

There is a preemptive Markovian repair process which can work on one broken machine at a time. This process is guided by a repair policy, which decides which of the currently broken machine to repair. If the policy chooses to service machine  $i$ , it incurs a cost of  $c_i$ , and machine  $i$  transitions back to the active state according to a memoryless random process with rate 1. (The time it takes to repair any machine is a random variable distributed as  $\text{Exponential}(1)$ ). The policy can interrupt a maintenance and begin working on another machine at any time. The goal is to design the repair policy that maximizes the infinite horizon time-average reward minus cost, which is the “value” of the policy.

For simplicity, we assume that time is continuous in the proceeding analysis. The memoryless and preemptive properties in our model imply that repair policies are Markovian, meaning that only information about the current state (i.e. the set of broken machines) matters, and not information such as residual service times. Such Markovian models are widely used for studying processor performance in multiprocessor systems [10].

Our problem can be viewed as a Markov Decision Process (MDP). In such problems, a policy chooses an action for every state, and this defines a Markov chain with specific values related to each node. The problem reduces to maximizing the total value of the Markov chain by optimizing over the policies. While dynamic programming algorithms find the optimal policy [2], the computation could take exponential time, and the policy could take exponential space to store since the number of states is  $2^n$ .

This leads to the question: Is there a simpler description of the optimal policy? In particular, is the optimal policy an “index” or priority policy of the following nature: compute an “index” for each machine based on its parameters and state, and repair the machine with the “index.” In our problem, since only broken machines need to be maintained, this corresponds to maintaining based on a fixed priority ordering. For a related and well-studied problem—the stochastic multi-armed bandits problem [2, 8, 18]—there is a simple and elegant optimal index policy termed the Gittins index policy [8]. The following example illustrates that no index policy can be optimal.

The following 3 machine example shows that no index policy is optimal; All  $c_i = 0$ , and  $(r_1, p_1) = (1, 1/4)$ ,  $(r_2, p_2) = (4, 1/4)$  and  $(r_3, p_3) = (7, 2/3)$ . The optimal policy, when some two machines are broken, gives machine 3 the highest priority, and machine 2 the next highest. But when all 3 machines are broken, the policy maintains machine 2. Since the relative order between machine 2 and 3 depends on the status of machine 1, this is not an index policy. This policy has value 7.4266, while the value of the best index policy (which has the priority order  $2 > 3 > 1$ ) is 7.4243.



This result is not surprising – a very similar problem (with rewards replaced by costs) is mentioned in [2] (vol. 2, ch. 1) as the simplest variant of the multi-armed bandits problem for which no index policy is optimal.

Though index policies are sub-optimal, they are desirable for their simplicity and ease of implementation. They are widely used as heuristics for the multi-class queuing and restless bandit problems [19,16,4]. For our problem, the next natural question is whether there is an index policy that is optimal?

The simplest and most natural priority scheme is the following: Repair the machine that maximizes the ratio between the expected gain and the expected repair time. The expected gain from machine  $i$  until it next breaks down is  $\frac{r_i}{p_i} - c_i$  and the expected repair time is 1. Therefore, this scheme corresponds to priority-ordering the machines in decreasing order of  $R_i = \frac{r_i}{p_i} - c_i$ . We term this the GREEDY policy. In addition to being intuitive, this policy naturally falls out of a LP relaxation for the problem (Section 2).

One immediate question is whether GREEDY is the optimal policy. Unfortunately, the GREEDY policy is not optimal even for two machines, where any policy is a priority scheme.

Consider two machines with  $c_1 = c_2 = 0$ ,  $(r_1, p_1) = (0.282 + \epsilon, 0.282)$  and  $(r_2, p_2) = (2.704, 2.704)$  ( $\epsilon \ll 1$ ). The GREEDY policy gives priority to machine 1 and has value 0.8, whereas prioritizing machine 2 yields value 0.856.

In the same vein, for the following three machine example, the optimal policy is not an index policy, and the optimal index policy is not the GREEDY policy: all  $c_i = 1$ ,  $(r_1, p_1) = (r_2, p_2) = (1, 1/4)$ , and  $(r_3, p_3) = (4, 3/2)$ .

Nevertheless, we show that the value of GREEDY is always within a factor of 1.51 of the optimal value. At the end of the paper, we use insights from our analysis to construct a bad example (Example 3), in which the optimal policy is more than 1.3 times better GREEDY. This shows that our 1.51 bound is reasonably sharp. The main contribution of this paper is in analyzing GREEDY through a numerical-based technique, which can exploit queuing structure more effectively than can conventional techniques.

In Section 2, we get a handle of the optimal policy by upper-bounding it using a simple linear program, which turns out to be the same as the LP that describes fractional knapsack. We reduce the problem to a program which has already been solved in a paper on stochastic knapsack [5,6], and this shows that GREEDY is a 2 approximation.

The resultant worst-case ratio from such conventional techniques is the same as that of machine replenishment problem [11,13], where a similar greedy scheme is shown to be a 2 approximation. Their analysis is based on rounding the solution to a convex program, which ignores the stochastic queuing interactions in our problem. In fact, our stochastic problem should be less “worst-case” since the optimal policy can not predict what exactly happens in the stochastic queuing interactions. But converting this intuition into an improved approximation ratio is non-trivial, since the active and inactive periods

for different machines interact via a complicated queuing process. (The GREEDY policy corresponds to a non-symmetric Markov chain with  $2^n$  nodes).

We proceed to prove a 1.51 approximation ratio using unconventional, computer-based techniques. In section 3, we construct an improved lower-bound for the contribution of each machine, taking into account its placement in the priority order and information about other machines. In section 4, we use this lower-bound show that the improved performance lower-bound for GREEDY (relative to the LP value) is the solution of a non-convex, non-linear optimization program with infinitely many variables. (In fact, it is equivalent to computing the shortest path in an infinite graph). Finally, we approximate the program using a computer, and rigorously prove that the error in approximation is small. This proves the 1.51 approximation ratio. We believe that our technique provides a general framework for optimizing lower-bounds of a type that is likely to arise in related stochastic optimization problems.

The stochastic machine maintenance problem is a closed multi-class queuing problem [3,11,17]. In general, there are several queues with servers, each of which provides service to packets in different service classes at different rates. When a packet completes service, it can transition to a different class and enter a different queue. The service completions are associated with costs and rewards. A policy in multi-class queuing specifies for each server which packet to receive service at each time instant. The machine replenishment problem can be modeled as a multi-class queuing problem as follows: Each machine is a “packet” and is associated with an “active” queue, where it receives service at rate  $p_i$  and reward at rate  $r_i$ . There is a common “broken” queue to which the packet transitions upon finishing service in its active queue. The server for the broken queue operates at rate 1 and a packet pays  $c_i$  upon finishing service. Since the multi-class queuing problems are in general PSPACE-hard to approximate [17], little is known in terms of analytic guarantees of natural policies for these problems. As far as we are aware, our results provide the first analysis of a natural greedy policy for such a problem.

The idea to derive a program whose solution is the approximation ratio is not new: “Factor-revealing Linear Programs” have been used for analyzing greedy algorithms for facility location [12], min-sum set cover [15], and online matching [14]. Since our program is non-linear and non-convex, we need our numerical machinery to solve for the approximation ratio.

The first part of our work is similar to the adaptivity gap proofs for greedy placement policies for stochastic knapsack by Dean, Goemans, and Vondrák [5,6]. Their analysis of the greedy algorithm uses a nice analytic/geometric argument to bound the worst case ratio of two functions. In fact, our 2 approximation analysis reduces to using exactly the same functions. But to yield a tighter bound, one needs to better exploit the queuing theoretic component of the problem and construct more complex bounding expressions. The second part of our work is similar to Goemans and Kleinberg’s approximation analysis [9] for the minimum latency problem. They also use a shortest path to compute their approximation

ratio, but since their expression for edge lengths is simple, they can compute their shortest path analytically. In contrast, the complexity of our bounding expressions necessitates our computational approach.

## 2 LP Bound and the 2 Approximation

**Definition 1.** Let  $t_i = \frac{p_i}{1+p_i}$  and  $R_i = \left(\frac{r_i}{p_i} - c_i\right)$ . □

- $t_i = \frac{p_i}{1+p_i}$
- $R_i = \left(\frac{r_i}{p_i} - c_i\right)$

As described in Section 1, the natural GREEDY policy prioritizes the machines in decreasing order of  $R_i$  and repairs the broken machine with the highest priority (provided the corresponding  $R_i > 0$ ). For simplicity of analysis, add a dummy machine with  $p = \infty, r = c = R = 0$ . Moreover, renumber the machines so that  $R_1 \geq R_2 \geq \dots \geq R_n$ . We upper-bound the value of the optimal policy by a linear program.

**Theorem 1.** □

$$\max \left\{ \sum_{i=1}^n R_i A_i \mid \sum_{i=1}^n A_i \leq 1, A_i \in [0, t_i] \right\}$$

$$LP = \sum_{i=1}^u t_i R_i + (1 - \sum_{i=1}^u t_i) R_v$$

where  $u = \max\{i \mid \sum_{j=1}^i t_j < 1\}$  and  $v = u + 1$ .  $R_v \geq 0$ .

Since our problem is a continuous time MDP, the optimal policy chooses a machine to maintain for each state, and this results in an ergodic Markov system. Consider any policy  $\mathcal{P}$ . Let  $A_i$  denote the long term average rate at which the policy maintains machine  $i$ . Since the policy can maintain one machine at a time,  $\sum_i A_i \leq 1$ .

If machine  $i$  is the only machine in the system, its repair completions define a renewal process with rate exactly  $\frac{1}{1+p_i} = t_i$  (refer to [7] for renewal theory). (The denominator comes from the fact that the expected service time and the expected active time are 1 and  $\frac{1}{p_i}$  respectively). The presence of other machines can only impede the maintenance process for machine  $i$ , so  $A_i \leq t_i$ . Therefore, the  $A_i$ 's in any policy  $\mathcal{P}$  satisfy the LP constraints.

Let  $U_i$  denote the probability (at a random time) that machine  $i$  is “active”. Since the machine transitions from “active” to “broken” in a memoryless fashion

and since the rate of transition to the broken state is the same as the rate of repair completions, we have  $A_i \times 1 = U_i p_i$ . Therefore, the reward of the policy per unit of time obtained from machine  $i$  is  $r_i U_i = A_i (r_i / p_i)$ , and the cost of maintenance is  $A_i (c_i)$ . Combining these equalities, the value of the policy is  $\sum_{i=1}^n R_i A_i$ , which is precisely the LP objective. Since we've shown that any policy  $\mathcal{P}$  is feasible for the LP, the solution to the LP upper-bounds the value of the optimal policy.

The LP encodes a fractional knapsack instance in which the knapsack capacity is 1, the profit per unit size of item  $i$  is  $R_i$ , and the size of item  $i$  is  $t_i$ . This implies that the LP solution is  $LP = \sum_{i=1}^u t_i R_i + (1 - \sum_{i=1}^u t_i) R_v$ .

Essentially, the LP relaxes the constraint that at most one machine is repaired at a time to requiring the  $\dots$  rate of repair to be at most 1, while allowing multiple repairs at once. The LP solution prioritizes maintenance for machines with the highest  $R$ 's, and this naturally leads to our GREEDY policy of prioritizing based on  $R_i$ 's. We now show that the GREEDY policy has an approximation ratio of at most 2 against the LP bound.

In the proof of Theorem 1, we defined  $A_i$  for every policy. From now on, we keep the notation but restrict the definition to the GREEDY policy. Moreover, we define some other key probabilities.

**Definition 2.**  $\dots$  GREEDY  $g_i, Q_i, \dots$

- $A_i = \dots$
- $g_i = \dots$
- $U_i = \dots$

**Lemma 1.**  $\dots$   $Q_i = \sum_{j=1}^{i-1} t_j$   $\dots$   $Q_1 = 0$   $\dots$   $g_i \geq 1 - Q_{i+1}$   $\dots$   $A_i \geq t_i(1 - Q_i)$

$\dots$  In the proof of Theorem 1, we showed that  $A_i \leq t_i$ . Thus, the total probability with which some machine from  $\{1, 2, \dots, i\}$  is being repaired,  $1 - g_i \leq \sum_{j=1}^i A_j \leq \sum_{j=1}^i t_j = Q_{i+1}$ . Rearranging,  $g_i \geq 1 - Q_{i+1}$ .

We now prove the second part. As we showed in the proof of Theorem 1,  $A_i = p_i U_i$ , which implies that  $U_i = \frac{1}{p_i} A_i$ . From part (1), we have  $g_{i-1} \geq 1 - Q_i$ . The GREEDY policy maintains machine  $i$  whenever machines  $1, 2, \dots, i - 1$  are broken and machine  $i$  is active, so  $A_i = g_{i-1} - g_i$ . Using these expressions and the obvious fact that  $\frac{g_i}{U_i} \leq 1$ , we have,

$$A_i \left(1 + \frac{1}{p_i}\right) \geq A_i \left(1 + \frac{g_i}{U_i} \frac{1}{p_i}\right) = A_i + g_i = g_{i-1} \geq 1 - Q_i \tag{1}$$

Since  $t_i = p_i / (p_i + 1)$ , the above inequality is equivalent to  $A_i \geq t_i(1 - Q_i)$ , which completes the proof.

Therefore, the value of the GREEDY policy,  $GREEDY = \sum_{i=1}^n A_i R_i \geq \sum_{i=1}^v t_i(1 - Q_i) R_i$ . Recall from Theorem 1 that the value of the optimal policy,  $OPT \leq LP = \sum_{i=1}^v t_i R_i$ . Our goal is to lower-bound the ratio  $GREEDY/LP$ . Note that

the presence of the dummy machine ensures that  $\sum_{i=1}^v t_i \geq 1$ , and if  $\sum_{i=1}^v t_i > 1$ , we can decrease *GREEDY/LP* by setting  $t_v = 1 - \sum_{i=1}^u t_i$ , because doing so decreases *GREEDY* while preserving *LP*. Thus, we can restrict ourselves to  $\sum_{i=1}^v t_i = 1$ . To prove the  $GREEDY/LP \geq 1$ , we just need:

$$\frac{\sum_{i=1}^v t_i(1 - Q_i)R_i}{\sum_{i=1}^v t_i R_i} \geq \gamma \quad \text{where} \quad t_i \geq 0, \quad \sum_{i=1}^v t_i = 1, \quad R_1 \geq R_2 \geq \dots$$

But this is equivalent to the program obtained for the start deadline model of stochastic knapsack [5], where it is shown via an elegant geometric argument that the maximum  $\gamma = 1/2$ . The proof also follows from applying the machinery in Section 4 to the function  $A(t, Q) = t(1 - Q)$ .

**Theorem 2.** *GREEDY*  $\geq 1/2$ .

### 3 Exploiting Queuing Structure in *GREEDY*

In the following sections, we sharpen our analysis and prove a better approximation ratio. We first use queuing theoretic arguments to construct a more intricate bound for the contribution of each machine (Theorem 3). The resulting expressions are too complex to be effectively analyzed through analytic techniques. In the section 4.2, we develop a numerical technique which computes the tightest possible overall bound given the per-machine bound that we prove here. We finally bound the error of this technique and show an approximation ratio of 1.51.

We begin by rearranging Inequality (II) from the proof of Lemma 1:

$$A_i \geq p_i \left( \frac{1 - Q_i}{p_i + g_i/U_i} \right) \tag{2}$$

In our approximation in Lemma 1, we used the weak bound  $g_i/U_i \leq 1$ . (Recall that  $g_i$  is the probability that the first  $i$  machines are active and  $U_i$  is the probability that machine  $i$  is active). We now show a tighter bound for the gap between  $g_i$  and  $U_i$ , and we exploit this to improve the approximation ratio.

**Lemma 2.** For  $i = 1, 2, \dots, v$ ,

$$\frac{g_i}{U_i} \leq \frac{1 + p_i}{1 + p_i + \sum_{j=1}^{i-1} t_j} \tag{3}$$

Let  $T_i$  be the probability that exactly one of the machines  $1, 2, \dots, i-1$  is broken and machine  $i$  is active. Call the state in which machines  $1, 2, \dots, i$  are all good  $G_i$ . The steady-state rate at which the Markov system defined by *GREEDY* transitions out of  $G_i$  is exactly  $g_i \times \sum_{j=1}^i p_j$ , and the rate at which the system transitions into  $G_i$  is  $(T_i + A_i) \times 1$ . Thus, by flow conservation,  $g_i \sum_{j=1}^i p_j = T_i + A_i$ .

Also, observe that  $T_i + g_i = U_i$ , and  $t_i = \frac{p_i}{1+p_i} \leq p_i$ . From the proof of Theorem 1,  $A_i = p_i U_i$ . Therefore,

$$g_i(p_i + \sum_{j=1}^{i-1} t_j) \leq g_i \sum_{j=1}^i p_j = T_i + A_i \leq U_i - g_i + p_i U_i$$

Separating out terms involving  $g_i$  and terms involving  $U_i$ , we obtain the desired inequality.

Plugging Equation (3) into Equation (2) and substituting  $t_i = \frac{p_i}{1+p_i}$ , we get

**Theorem 3.**  $A_i \geq t_i(1 - Q_i) \frac{1+Q_i(1-t_i)}{1+t_i Q_i(1-t_i)}$   $\dots$   $A(t, Q) = t(1 - Q) \frac{1+Q(1-t)}{1+tQ(1-t)}$   $\dots$   $A_i \geq A(t_i, Q_i)$

Unlike the previous bound  $A_i \geq t_i(1 - Q_i)$ , the above expression is non-linear and non-convex, so we need a new technique to effectively exploit this bound.

### 4 Numerically Computing the Approximation Ratio

We now show a general technique for numerically computing (and rigorously proving) the best possible approximation ratio that can be obtained from the bounding expressions shown in Theorem 3. In fact, our machinery will work with any expression  $A(t, Q)$  that satisfies the conditions stated below.

**Lemma 3.**  $\dots$   $A(t, Q) \dots$   $\square \dots t, Q \in [0, 1]$

$$\begin{aligned}
 & A(t, Q) \in [0, t] \quad \dots \quad A(0, Q) = 0 \quad \dots \quad Q \in [0, 1] \\
 & \frac{\partial A}{\partial t} \geq 0 \quad \dots \quad \frac{\partial A}{\partial Q} \leq 0 \\
 & \dots \quad F(t, Q) = \frac{A(t, Q)}{t} \quad \dots \quad \frac{\partial F}{\partial t} \leq 0 \quad \dots \quad \frac{\partial F}{\partial Q} \leq 0 \\
 & \dots \quad C
 \end{aligned}$$

**Observation 1.**  $\dots$   $\left| \frac{\partial^2 A}{\partial Q^2} \right| \leq \frac{5}{8} \quad \left| \frac{\partial^2 A}{\partial t Q} \right| \leq \frac{15}{4} \quad \left| \frac{\partial^2 A}{\partial t^2} \right| \leq 6$

The above conditions are fairly intuitive and reasonable. Since  $t_i = \frac{p_i}{1+p_i}$ , condition 1 simply states that machines with 0 breakdown rates have 0 maintenance rates. Condition 2 states that everything else being equal, the higher the breakdown rate, the higher the maintenance rates. Moreover, for a particular machine, the “greater amount” of machines with higher priority, the more that machine has to wait for other maintenances, and the lower the maintenance rate. In condition 3,  $F$  measures how well the real maintenance rate  $A$  keeps up with the maintenance rate of the LP, in which more than one machine can be maintained simultaneously. It’s reasonable to expect that the larger the breakdown rate of the machine or the “greater amount” of machines with higher priority, the more

“handicapped” is GREEDY because of the one-repair-at-a-time constraint, and  $F$  decreases.

**Observation 2.**  $\sum_{i=1}^v A(t_i, Q_i) R_i \geq t_v \cdot (1 - \sum_{i=1}^{v-1} t_i) \cdot \text{GREEDY} \geq \sum_{i=1}^v t_i > 1$   
 $\gamma \leq \text{GREEDY}/LP \leq \sum_{i=1}^v t_i \leq 1$

**4.1 Reduction to a Shortest Path Problem**

Using the LP-bound proved in Theorem 1 and Observation 2, we see that the inverse of the best approximation ratio that can be proven from Theorem 3, denote by  $\gamma$ , is the solution of the following optimization problem.

Compute  $\gamma = \inf_{v \in \mathcal{N}} \mathcal{H}(v)$ , where the infimum is taken over all positive integer  $v$ , and  $\mathcal{H}(v)$  is the solution of the following non-linear program:

$$\text{Minimize } \left( \frac{\sum_{i=1}^v A(t_i, Q_i) R_i}{\sum_{i=1}^v t_i R_i} \right) \quad \text{s.t.} \quad \begin{aligned} \sum_{i=1}^v t_i &\leq 1, & t_i &\in (0, 1] \quad \forall i \\ R_i &\geq R_{i+1} & \forall i &= 1, \dots, v-1 \\ Q_{i+1} &= Q_i + t_i & \forall i &= 1, \dots, v \\ Q_1 &= 0, & R_v &> 0 \end{aligned}$$

We insist that  $R_v > 0$  because it is not worth repairing machines with  $R \leq 0$ . We now show how to compute  $\gamma$  to any degree of accuracy. First, we show that the solution of Problem 2 is the same as the solution to the following shortest path problem:

For every  $x, y \in [0, 1]$  with  $x \leq y$ , define a directed edge from  $x$  to  $y$  with length  $\mathcal{L}(x, y) = A(y - x, x) \geq 0$ . Note that  $\mathcal{L}(x, x) = 0$  for all  $x$ . These edges define a graph on the number line  $[0, 1]$ . For positive integer  $v$ , define  $\mathcal{D}_v(x, y)$  as the length of the shortest path from  $x$  to  $y$  using  $v$  edges. Define  $\mathcal{D}(x, y) = \inf_v \mathcal{D}_v(x, y)$ . Compute  $\mathcal{D}(0, 1)$ .

Note that  $\mathcal{D}_v(0, 1)$  is the same as the following program, which is obtained by setting all  $R_i = 1$  and  $\sum_{i=1}^v t_i = 1$  in Problem 2

$$\text{Minimize } \sum_{i=1}^v A(t_i, Q_i) \quad \text{s.t.} \quad \begin{aligned} \sum_{i=1}^v t_i &= 1, & t_i &\in [0, 1] \quad \forall i \\ Q_{i+1} &= Q_i + t_i & \forall i &= 1, 2, 3, \dots, v \\ Q_1 &= 0, & Q_i &\in [0, 1] \quad \forall i \end{aligned}$$

The next lemma is crucial for showing the equivalence of Problems 2 and 3.

**Lemma 4.**  $L \in (0, 1] \quad \frac{\mathcal{D}(0, L)}{L} \geq \mathcal{D}(0, 1)$

We will show that  $\frac{\mathcal{D}_k(0, L)}{L} \geq \mathcal{D}_k(0, 1)$  for all  $k \geq 1$ . The lemma will follow by taking the infimum. For fixed  $L, k$ , suppose the shortest path from 0 to  $L$  is  $0 = Q_1 \leq Q_2 \leq \dots \leq Q_{k+1} = L$ . Recall  $F(t, Q) = A(t, Q)/t$  from Lemma 3. Since  $t_j = Q_{j+1} - Q_j$ ,  $\mathcal{D}_k(0, L) = \sum_{j=1}^k A(Q_{j+1} - Q_j, Q_j) = \sum_{j=1}^k A(t_j, Q_j) = \sum_{j=1}^k t_j F(t_j, Q_j)$ .

By Lemma 3,  $F(t, Q)$  is non-increasing in  $t, Q$ , so

$$\frac{\mathcal{D}_k(0, L)}{L} = \sum_{j=1}^k \frac{t_j}{L} F(t_j, Q_j) \geq \sum_{j=1}^k \frac{t_j}{L} F\left(\frac{t_j}{L}, \frac{Q_j}{L}\right) = \sum_{j=1}^k A\left(\frac{Q_{j+1}}{L} - \frac{Q_j}{L}, \frac{Q_j}{L}\right)$$

Since  $\sum_{j=1}^k t_j = Q_{k+1} = L$ , the RHS of the above inequality is the length of a path with  $k$  edges from 0 to 1 passing through the points  $0 = \frac{Q_1}{L} \leq \frac{Q_2}{L} \leq \dots \leq \frac{Q_{k+1}}{L} = 1$ . Therefore, the RHS is at least  $\mathcal{D}_k(0, 1)$ .

**Theorem 4.**  $\gamma \geq \mathcal{D}(0, 1)$  i.e.  $\gamma = \mathcal{D}(0, 1)$

In the optimal solution for  $\mathcal{H}(v)$  in Problem 2, make the substitution  $R_i = \sum_{j=i}^v x_j$  where  $x_j \geq 0$ . We can do this because the  $R_i$ 's are non-increasing.

$$\begin{aligned} \mathcal{H}(v) &= \frac{\sum_{i=1}^v \left( \left( \sum_{j=i}^v x_j \right) A(t_i, Q_i) \right)}{\sum_{i=1}^v t_i \left( \sum_{j=i}^v x_j \right)} = \frac{\sum_{j=1}^v x_j \left( \sum_{i=1}^j A(Q_{i+1} - Q_i, Q_i) \right)}{\sum_{j=1}^v x_j \left( \sum_{i=1}^j t_i \right)} \\ &\geq \frac{\sum_{j=1}^v x_j \mathcal{D}(0, Q_{j+1})}{\sum_{j=1}^v x_j Q_{j+1}} \geq \min_{j=1}^v \frac{\mathcal{D}(0, Q_{j+1})}{Q_{j+1}} \geq \mathcal{D}(0, 1) \end{aligned}$$

The last inequality, which is the crux of the proof, follows from Lemma 4. All quantities are well-defined since  $Q_{j+1} > 0$  for  $j \geq 1$ , and  $x_v > 0$ . Therefore,  $\gamma \geq \mathcal{D}(0, 1)$ . Now, setting all  $R_i = 1$  and  $Q_{v+1} = 1$ , we obtain from the minimality condition on  $H(v)$  that  $\mathcal{H}(v) \leq \mathcal{D}_v(0, 1)$ . This implies that  $\gamma \leq \inf_v \mathcal{D}_v(0, 1) = \mathcal{D}(0, 1)$ . These two inequalities imply that  $\gamma = \mathcal{D}(0, 1)$ .

### 4.2 Solving Problem 3: A Computer-Based Technique and Its Analysis

Theorem 4 implies that the inverse of the desired approximation ratio is the solution to Problem 3. But this is still not easy to find since Problem 3 is still a huge non-linear, non-convex optimization problem with non-separable objective. As a natural first step, we consider a discrete version of the problem.

For any positive integer parameter  $N$ , call  $x \in [0, 1]$  a ‘‘lattice point’’ if  $x = i/N$  for some integer  $i$ . For every pair of lattice points  $x, y \in [0, 1]$  with  $x < y$ , define a directed edge from  $x$  to  $y$  with length  $\mathcal{L}(x, y) = A(y - x, x + \frac{1}{N}) \geq 0$ . (The new lengths are smaller by Lemma 3 and this change is crucial for the proof to work). These edges define a graph over the set of lattice points in  $[0, 1]$ . Compute  $\gamma_N^*$ , the length of the shortest path from 0 to 1 in this graph. The key difference from Problem 3 is the restriction of the path to lattice points, and the change of the length of the  $(x, y)$  edge from  $A(y - x, x)$  to  $A(y - x, x + \frac{1}{N})$

Since the corresponding graph is directed and acyclic, Problem 4 can be solved in  $O(N^2)$  time on a computer. Figure 2(a) plots  $\gamma_N^*$  versus  $\log N$  for  $N \leq 10^5$ . The computation shows  $\gamma_N^* \geq 0.6626$  for  $N = 10^5$ . Since 64-bit precision was



used in the computer program, and the number of edges lengths summed up in getting the final result is at most  $N$ , the error introduced due to the finite precision of the computer is negligible.

The question now is: Using the conditions stated in Lemma 3 it is easy to show  $\gamma \leq \gamma_N^* + \frac{1}{N}$ . However, we need a bound in the other direction. The key contribution in this section is the next theorem, showing that even for small  $N$ , the computed value  $\gamma_N^*$  indeed implies a good lower bound on  $\gamma$ :

**Theorem 5.**  $A(t, Q) = t(1 - Q) \frac{1+Q(1-t)}{1+tQ(1-t)}$ ,  $C = 6$ ,  $\gamma, \mathcal{D} \geq \gamma_N^* - \frac{5C}{2N}$

$$\gamma = \mathcal{D}(0, 1) \geq \gamma_N^* - \frac{5C}{2N}$$

In particular, for  $A(t, Q) = t(1 - Q) \frac{1+Q(1-t)}{1+tQ(1-t)}$ , recall from Observation 1 in Section 4 that  $C = 6$ . Using  $N = 10^5$ , we compute  $\gamma_N^* \geq 0.6626$ , and the above theorem shows that  $\gamma \geq 0.6626 - 0.0002 = 0.6624$ , which shows an upper-bound of 1.51 on the approximation ratio. The rest of this section is devoted to proving Theorem 5.

The overall proof idea is to construct a feasible solution for Problem 4 from the optimal solution to  $\mathcal{D}_v(0, 1)$ . We show that this feasible solution has a value at most  $2.5C/N$  larger than  $\mathcal{D}_v(0, 1)$ , which implies  $\gamma_N^* \leq \mathcal{D}_v(0, 1) + 2.5C/N$ . Taking the infimum on  $v$  proves Theorem 5.

There are two main ideas in the construction: The first is to move certain points on the shortest path  $\mathcal{D}_v(0, 1)$  to the closest lattice point, while preserving what we call the “local-minimum” invariant: each non-lattice point lies on a shortest path containing its immediate neighbors. This invariant allows us to bound the increase in path length, caused by moving points, in terms of the square of the distance moved, instead of the distance moved, which is crucial to our proof that the overall path-length increase is inversely related to  $N$ . The second idea is that moving every node to a lattice location increases the path length too much if  $v \gg N$ . To overcome this problem, we only move a few chosen points, and then use the slack from the new edge lengths defined in Problem 4 to delete the remaining points, without increasing the total path length. We present these two ideas in reverse order. In the proofs, assume fixed  $v, N$ .

We first define an intermediate path type which has more structure than a general path (Problem 3), but is not as constrained as a lattice path (Problem 4). This path can use non-lattice points, but for each non-lattice point, the path contains the lattice points to the immediate left and right.

**Definition 1.**  $P$  semi-discrete,  $0 = Q_1 \leq Q_2 \leq \dots \leq Q_{v+1} = 1$ ,  $\forall Q \in P \implies \frac{i}{N} \in P, \frac{i+1}{N} \in P$

$$\sum_{j=1}^v A(Q_{j+1} - Q_j, Q_j)$$

**Lemma 5.**  $\dots \gamma_N^*$

Let semi-discrete path  $P_{sd}$  correspond to points  $0 = Q_1 \leq Q_2 \leq \dots \leq Q_{v+1} = 1$ . The length of this path is  $W = \sum_{j=1}^v A(Q_{j+1} - Q_j, Q_j)$ . Removing the non-lattice points, we form a discrete lattice path  $P_{dl}$ , which passes through lattice-points  $0 = Q_{i_1} < Q_{i_2} < \dots < Q_{i_{k+1}} = 1$ . This new path is a feasible for Problem 4 and has length  $Z = \sum_{j=1}^k A(Q_{i_{j+1}} - Q_{i_j}, Q_{i_j} + \frac{1}{N})$ . It suffices to show that  $Z \leq W$ . Consider two consecutive points  $Q_a < Q_b$  on  $P_{dl}$ . These are also points (not necessarily consecutive) in  $P_{sd}$ .

**Case 1.** Suppose  $Q_b > Q_a + 1/N$ . Then,  $Q_a < Q_b$  are not consecutive lattice points, which implies that they are consecutive points on  $P_{sd}$ . Now,  $A(Q_b - Q_a, Q_a) \geq A(Q_b - Q_a, Q_a + \frac{1}{N})$  by Lemma 3. The former term is the contribution of  $(Q_a, Q_b)$  to  $W$  and the latter (which is smaller) is the contribution to  $Z$ .

**Case 2.** Suppose  $Q_b = Q_a + 1/N$ , then there could be intermediate non-lattice points in  $P_{sd}$ . Suppose  $Q_a = i/N$  and  $Q_b = (i + 1)/N$  for integer  $i$ . Let the path segment in  $P_{sd}$  contain points  $\frac{i}{N} = Q_a \leq Q_{a+1} \leq \dots \leq Q_b = \frac{i+1}{N}$ . The contribution of this path segment to  $W$  is (letting  $\Delta_j = Q_{j+1} - Q_j$ ):

$$\sum_{j=a}^{b-1} \Delta_j F(\Delta_j, Q_j) \geq \sum_{j=a}^{b-1} \Delta_j F\left(\frac{1}{N}, Q_a + \frac{1}{N}\right) = \frac{1}{N} F\left(\frac{1}{N}, Q_a + \frac{1}{N}\right)$$

where the inequality comes from the fact that  $F(t, Q)$  is decreasing in both variables (Lemma 3). The contribution of  $(Q_a, Q_b)$  to  $Z$  is (set  $\Delta = Q_b - Q_a$ ):

$$A(\Delta, Q_a + \frac{1}{N}) = A(\Delta, Q_b) = \Delta \cdot F(\Delta, Q_b) = \frac{1}{N} F\left(\frac{1}{N}, Q_a + \frac{1}{N}\right)$$

Summing the bounds for every segment  $(Q_a, Q_b)$ , we get  $Z \leq W$ , which proves the lemma.

We now describe an algorithm that converts the absolute shortest path (the solution to Problem 3) to a semi-discrete path whose length is at most  $2.5C/N$  larger. (The algorithm is only a theoretical argument and is not intended to be run). Combined with Lemma 5, this proves Theorem 5. Suppose that the optimal path that yields  $\mathcal{D}_v(0, 1)$  is  $P$ , with points  $0 = Q_1 \leq Q_2 \leq \dots \leq Q_{v+1} = 1$ . The lengths of edges are the original lengths defined in Problem 3. We run the algorithm presented in Figure 1, which transforms the optimal path  $P$  into a semi-discrete path  $P'$ .

We now bound the increase in path lengths from transforming  $P$  to  $P'$ . Note that step (c) cannot increase the path length, since it replaces a path segment by the corresponding shortest path. The only step that can increase the path length is Step (b), which is executed at most  $N$  times. We can bound the increase in

**Converting optimal path  $P$  of Problem 3 to a semi-discrete path  $P'$ .**

**For  $i = 1$  to  $N$ :**

1. **If  $\frac{i-1}{N} \notin P$  and  $\exists Q_j \in P$  s.t.  $Q_j \in (\frac{i-1}{N}, \frac{i}{N})$  then:**
  - (a) Let  $j^* = \operatorname{argmin} \{Q_j \in P, Q_j \in (\frac{i-1}{N}, \frac{i}{N})\}$ .
  - (b)  $Q_{j^*} \leftarrow \frac{i-1}{N}$ .
  - (c) Replace segment  $\{Q_{j^*}, Q_{j^*+1}, \dots, Q_{v+1}\}$  in  $P$  with  $\mathcal{D}_{v-j^*+1}(Q_{j^*}, Q_{v+1})$
2. **If  $\frac{i}{N} \notin P$  and  $\exists Q_j \in P$  s.t.  $Q_j \in (\frac{i-1}{N}, \frac{i}{N})$  then:**
  - (a)  $j^* = \operatorname{argmax} \{Q_j \in P, Q_j \in (\frac{i-1}{N}, \frac{i}{N})\}$ .
  - (b)  $Q_{j^*} \leftarrow \frac{i}{N}$ .
  - (c) Replace segment  $\{Q_{j^*}, Q_{j^*+1}, \dots, Q_{v+1}\}$  in  $P$  with  $\mathcal{D}_{v-j^*+1}(Q_{j^*}, Q_{v+1})$

**Fig. 1.** Algorithm for making the optimal path  $\mathcal{D}_v(0, 1)$  semi-discrete

each execution by exploiting the local-minimum invariant that is enforced by Step (c). This invariant states that just before executing Step (b),  $Q_{j^*}$  lies on the shortest path between the two adjacent points in  $P$ .

**Lemma 6.**

Suppose the point being moved in Step (b) is  $Q_j$ . Let the adjacent points to  $Q_j$  in  $P$  just before the step be  $Q_{j-1}, Q_{j+1}$ . Assume w.l.o.g,  $Q_{j-1} < Q_j < Q_{j+1}$ . Since  $Q_{j-1}, Q_j, Q_{j+1}$  are consecutive nodes on the shortest path segment to  $Q_{v+1}$ ,  $Q_j$  must equal to the  $x \in (Q_{j-1}, Q_{j+1})$  that minimizes the sum of lengths of edges  $(Q_{j-1}, x)$  and  $(x, Q_{j+1})$ .  $x = Q_j$  minimizes  $f(x) = A(x - Q_{j-1}, Q_{j-1}) + A(Q_{j+1} - x, x)$ . This means that  $f'(Q_j) = 0$ . Now,

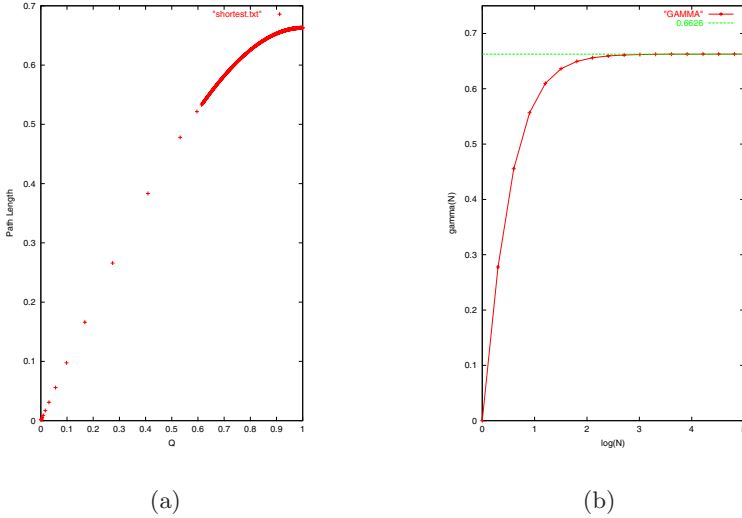
$$f''(x) = \frac{\partial^2}{\partial t^2} A(x - Q_{j-1}, Q_{j-1}) + \frac{\partial^2}{\partial t^2} A(Q_{j+1} - x, x) - 2 \frac{\partial^2}{\partial t \partial Q} A(Q_{j+1} - x, x) + \frac{\partial^2}{\partial Q^2} A(Q_{j+1} - x, x)$$

Since the second partials of  $A(t, Q)$  are bounded in absolute value by  $C$ ,  $|f''(x)| \leq 5C$ . By Taylor's expansion,  $|f(Q_j \pm d) - f(Q_j)| \leq 5Cd^2/2$ . Finally note that the change in the length of  $P$  in Step (b) is  $|f(Q_j \pm d) - f(Q_j)|$ , where  $d \leq \frac{1}{N}$ . The desired result follows.

(of Theorem 5) Start with the path corresponding to  $\mathcal{D}_v(0, 1)$ . Hypothetically apply the algorithm in Figure 1 to make the path semi-discrete. By applying Lemma 6 at most  $N$  times, we see that the algorithm increases the path length by at most  $2.5C/N$ . By Lemma 5, the length of this new semi-discrete path is at least  $\gamma_N^*$ . Therefore,  $\gamma_N^* \leq \mathcal{D}_v(0, 1) + 2.5C/N$ . Taking the infimum on  $v$  proves the theorem.

Applying Theorem 5 computationally with  $N = 10^5$  shows the following:

**Theorem 6.** GREEDY  $\gamma_N^* \geq 1.51$



**Fig. 2.** (a) The optimal path  $\mathcal{D}(0,1)$  for  $s_i = 1$ . The horizontal distance between consecutive points represents  $t_i$ , and the vertical distance the edge length. (b) Plot of  $\gamma_N^*$  as a function of  $\log_{10} N$ .

The shortest path found using the computer has an interesting structure: As plotted in Figure 2(b), the values of  $t_i$  initial scale geometrically. We imitate this scaling and construct the following worst-case example, which shows a lower bound of 1.3 on the approximation ratio of GREEDY against the LP. This makes our 1.51 approximation fairly sharp.

Consider 6 machines with  $t_1 = t_2 = 1/32, t_3 = 1/16, t_4 = 1/8, t_5 = 1/4$ , and  $t_6 = 1/2$ . Set  $p_i = \frac{t_i}{1-t_i}$ . The  $r_i$  are set so that  $R_1 = R_2 + \epsilon$ , and so on. The ratio between the LP bound and the GREEDY policy is approximately 1.3.

The technique for described in this section applies to any problem that requires bounding the sum of some function  $A(y_i, z_i)$ , where  $z_j = \sum_{i=1}^{j-1} y_i$  and  $A(y, z)$  satisfies the reasonable conditions described in Lemma 3. We think that this kind of a setup is general enough to appear in related queuing problems, so our technique might help sharpen other approximation analyses.

**Acknowledgment.** We thank Shivnath Babu, Jen Burge, and Sudipto Guha for their helpful suggestions.

## References

1. Bar-Noy, A., Bhatia, R., Naor, J., Schieber, B.: Minimizing service and operation costs of periodic scheduling. In: Proc. ACM-SIAM Symp. on Discrete Algorithms, pp. 11–20 (1998)
2. Bertsekas, D.: Dynamic Programming and Optimal Control. 2nd edn. Athena Scientific (2001)

3. Bertsimas, D., Gamarnik, D., Tsitsiklis, J.: Performance of multiclass markovian queueing networks via piecewise linear Lyapunov functions. *Annals of Applied Probability* 11(4), 1384–1428 (2002)
4. Bertsimas, D., Niño-Mora, J.: Restless bandits, linear programming relaxations, and a primal-dual index heuristic. *Oper. Res.* 48(1), 80–90 (2000)
5. Dean, B., Goemans, M., Vondrák, J.: Approximating the stochastic knapsack problem: The benefit of adaptivity. In: *Proc. of the 2004 Annual Symp. on Foundations of Computer Science* (2004)
6. Dean, B., Goemans, M., Vondrák, J.: Adaptivity and approximation for stochastic packing problems. In: *Proc. of the 2005 Annual ACM-SIAM Symp. on Discrete Algorithms* (2005)
7. Durrett, R.: *Probability: Theory and Examples*, 3rd edn. Duxbury Press (2004)
8. Gittins, J.C., Jones, D.M.: A dynamic allocation index for the sequential design of experiments. *Progress in statistics (European Meeting of Statisticians)* (1972)
9. Goemans, M.X., Kleinberg, J.M.: An improved approximation ratio for the minimum latency problem. In: *SODA*, pp. 152–158 (1996)
10. Goseva-Popstojanova, K., Trivedi, K.S.: Stochastic modeling formalisms for dependability, performance and performability. In: *Performance Evaluation: Origins and Directions*, pp. 403–422 (2000)
11. Harrison, J.M.: Dynamic scheduling of a multiclass queue: Discount optimality. *Operations Research* 23(2), 370–382 (1975)
12. Jain, K., Mahdian, M., Saberi, A.: A new greedy approach for facility location problems. In: *Proc. of the 2002 Annual ACM Symp. on Theory of Computing* (May 2002)
13. Kenyon, C., Schabanel, N.: The data broadcast problem with non-uniform transmission times. In: *Proc. ACM-SIAM Symp. on Discrete Algorithms* (1999)
14. Mehta, A., Saberi, A., Vazirani, U., Vazirani, V.: Adwords and generalized on-line matching. In: *FOCS 2005. Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pp. 264–273 (2005)
15. Munagala, K., Babu, S., Motwani, R., Widom, J.: The pipelined set cover problem. In: *Proc. Intl. Conf. Database Theory* (2005)
16. Niño Mora, J.: Restless bandits, partial conservation laws and indexability. *Adv. in Appl. Probab.* 33(1), 76–98 (2001)
17. Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of optimal queuing network control. *Math. Oper. Res.* 24(2), 293–305 (1999)
18. Tsitsiklis, J.N.: A short proof of the Gittins index theorem. *Annals of Applied Probability* 4(1), 194–199 (1994)
19. Whittle, P.: Restless bandits: Activity allocation in a changing world. *Appl. Prob.* 25(A), 287–298 (1988)

# A Polynomial Time Approximation Scheme for the Square Packing Problem

Klaus Jansen<sup>1,\*</sup> and Roberto Solis-Oba<sup>2,\*\*</sup>

<sup>1</sup> Institut für Informatik  
Universität zu Kiel, Kiel, Germany  
kj@informatik.uni-kiel.de

<sup>2</sup> Department of Computer Science  
The University of Western Ontario, London, Canada  
solis@csd.uwo.ca

**Abstract.** Given a set  $Q$  of squares with positive profits, the *square packing problem* is to select and pack a subset of squares of maximum profit into a rectangular bin  $\mathcal{R}$ . We present a polynomial time approximation scheme for this problem, that for any value  $\epsilon > 0$  finds and packs a subset  $Q' \subseteq Q$  of profit at least  $(1 - \epsilon)OPT$ , where  $OPT$  is the profit of an optimum solution. This settles the approximability of the problem and improves on the previously best approximation ratio of  $5/4 + \epsilon$  achieved by Harren's algorithm.

## 1 Introduction

Let  $Q = \{Q_1, Q_2, \dots, Q_n\}$  be a set of squares with positive profits. For a square  $Q_i$ , its size  $s_i$  is the length of one of its sides, and its profit is denoted as  $p_i$ . The problem is to select and pack a maximum profit subset of squares into a given rectangular bin  $\mathcal{R}$ . We consider only orthogonal packings, i.e. packings in which the sides of the squares are parallel to the sides of the bin. Geometric packing problems, like this one, have received a lot of attention recently from the algorithms community [1,6,8], as these problems have numerous applications in stock cutting, VLSI design, advertisement placement, image processing, and scheduling [5,6,8]. Furthermore, some of these problems are fundamental geometric problems that have been extensively studied by the Discrete Geometry community [7,11].

The square packing problem is known to be strongly NP-hard even for the restricted case of packing squares with unit profits [10]. For the version of the problem with unit profits, Jansen and Zhang [6] designed a polynomial time approximation scheme (PTAS). Fishkin et al. [4] studied the so-called square packing problem with resource augmentation and designed an algorithm that

---

\* Research supported in part by the EU project AEOLUS contract number 015964 and by the DAAD German Academic Exchange Service.

\*\* Research supported in part by the Natural Science and Engineering Research Council of Canada grant R3050A01.

packs a subset of squares of profit at least  $(1 - \epsilon)OPT$  into an augmented square bin  $[0, 1 + \epsilon] \times [0, 1 + \epsilon]$ , where  $OPT$  is the maximum profit of any subset of squares that can be packed into a unit size square bin. They also gave a PTAS for the problem without resource augmentation for the case when  $p_i = s_i^2$  for every  $Q_i \in Q$ . For the square packing problem with arbitrary profits, the previously best algorithm, by Harren, [5] has performance ratio  $\frac{5}{4} + \epsilon$ , for any  $\epsilon > 0$ .

We improve upon Harren's algorithm by presenting a PTAS for the square packing problem that selects and packs in  $\mathcal{R}$  squares of profit at least  $(1 - \epsilon)OPT$ , where  $OPT$  is the optimum profit and  $\epsilon > 0$  is any given positive constant. This is the best possible algorithm for the problem in the sense that the square packing problem is strongly NP-hard [10] and, so, it does not admit a fully polynomial time approximation scheme (FPTAS) unless  $P = NP$ .

For presentation simplicity, we only describe our algorithm for the case when  $\mathcal{R}$  is a unit size square bin. In the full version of the paper we will show how to deal with the case of a rectangular bin. The description is divided in two parts. First, we show a series of transformations that simplify the structure of an optimum solution  $P^*$  while only slightly decreasing the total profit of the squares packed in the bin. The transformations are based on the intuitive observation that large and high profit squares must be accurately positioned in the bin so they all fit; however, small, low profit squares might be positioned less carefully, as small mistakes in their positioning only causes a small loss in profit due to some of them not fitting in the bin. This observation has been used in the past for solving other packing and scheduling problems. One crucial point of our work that is different from past research is that we cannot round up the dimensions of large squares, as then we might need to increase the size of the bin to be able to pack the enlarged squares. This makes our problem a lot harder than problems that allow resource augmentation.

We define a class of canonical packings for the relatively small set of high profit squares in a feasible solution; these packings leave free space in the bin that can be split into a small set of rectilinear polygonal regions. We consider three types of these regions: large, elongated, and small. A large region is rectangular and its two dimensions are much larger than the sizes of the squares packed in it, so a simple shelf-packing algorithm, like NFDS [2], can pack squares of nearly optimum profit there. An elongated region is also rectangular, but it has one dimension much larger than the other, so, a strip packing algorithm can be used to pack squares there to near optimality.

Packing squares in a small region is considerably more difficult though, as there might not be enough room to pack any squares crossing the borders of a small region. We handle this situation by allowing some of the regions to be merged together. The drawback of doing this is that these regions get more complex shapes, and squares of large size relative to the dimensions of these regions must be packed very carefully. We deal with these regions by applying our transformations recursively on them. This creates a hierarchical organization for the regions, and we show that regions belonging to a constant number of levels

at the top of this hierarchy store squares of nearly optimum profit. This allows us to bound the number of required recursive applications of the transformations.

The result of the transformations is a near optimum solution  $P^+$  with a very regular structure: The packing  $P^+$  can be divided into a constant number of rectangular regions; in each region squares are packed either using the greedy NFDS algorithm, or groups of squares of similar sizes are packed in strips of the same width. Showing that a near-optimum packing with this particular structure exists is the core of our work as it allows a simple enumeration-based algorithm to compute a solution of profit close to that of  $P^+$ . The transformations are rather complex and they are described in Sections 2-5.

The second part of our algorithm is described in Section 7 and it shows how to build a very large, but polynomial in  $n$ , number of possible solutions with the same structure as  $P^+$ . We show that one of these solutions has profit very close to the profit of  $P^+$ , thus proving the existence of a PTAS for the square packing problem. The main contribution of this work is to settle the approximability of the square packing problem.

## 2 Big and Small Squares

Let  $\epsilon > 0$  be the required precision and, without loss of generality, let  $1/\epsilon$  be integer and  $\epsilon \leq 1/3$ . Fix an instance  $Q$  of the square packing problem and an optimum solution  $P^*$  for it. Let  $Q^*$  be the set of squares selected by  $P^*$  and let  $OPT$  be the profit of  $P^*$ . We can assume that  $Q^*$  has more than  $1/\epsilon$  squares, as otherwise we could find  $Q^*$  in  $O(1)$  time by simply enumerating all subsets of at most  $1/\epsilon$  squares from  $Q$  and then selecting the subset  $Q'$  of largest profit that can be packed in the bin using the upper-left justified packing algorithm of Section 7.

As mentioned above, our algorithm deals differently with small and large squares. Thus, we first need to define the notion of “small” and “large”. Define  $\rho_0 = 1$ , and  $\rho_k = \rho_{k-1}^4 (\epsilon/4)^{5+2^{k-1}}$  for all integers  $k \geq 1$ , where  $\eta = \log_{1-\epsilon} \epsilon$ . Then, it is not hard to see that  $\rho_k = (\epsilon/4)^{(5+2^{\eta+1})(4^k-1)/3}$  for all  $k \geq 0$ .

For each integer  $k \geq 1$  we define  $\mathcal{Q}_k = \{Q_i \in Q \mid s_i \in (\rho_k, \rho_{k-1}]\}$ . Let  $\tau$  be the smallest index such that the total profit of the squares in  $\mathcal{Q}_\tau^* = \mathcal{Q}_\tau \cap Q^*$  is at most  $\epsilon OPT$ . Observe that  $\tau \leq 1/\epsilon$ .

Partition  $Q$  into three groups: the large squares  $\mathcal{B} = \{Q_i \in Q \mid s_i > \rho_{\tau-1}\}$ , the medium squares  $\mathcal{M} = \{Q_i \in Q \mid \rho_\tau < s_i \leq \rho_{\tau-1}\}$ , and the small squares  $\mathcal{S} = \{Q_i \in Q \mid s_i \leq \rho_\tau\}$ . Similarly, define  $\mathcal{B}^* = \mathcal{B} \cap Q^*$ ,  $\mathcal{M}^* = \mathcal{M} \cap Q^*$ , and  $\mathcal{S}^* = \mathcal{S} \cap Q^*$ . Note that  $\mathcal{M}^* = \mathcal{Q}_\tau^*$ , so the total profit of the medium squares in  $\mathcal{M}^*$  is at most  $\epsilon OPT$ . In the next sections we show how to modify  $P^*$  to produce a near-optimum solution  $P^+$  with a regular structure.

To construct  $P^+$  we need to consider three cases: (i) when  $\mathcal{B}^* = \emptyset$ , (ii) when every big square has profit larger than  $\epsilon OPT$ , and (iii) when at least one big square has profit no larger than  $\epsilon OPT$ . Case (ii) is the most complex since, as we will show, it requires us to recursively partition the bin into successively smaller blocks where squares are re-classified as big, medium, and small depending on their relative sizes with respect to the dimensions of the blocks.



### 3 Instances without Big Squares

If  $\mathcal{B}^* = \emptyset$ , then we just remove  $\mathcal{M}^*$  from  $P^*$  to get a solution  $P^+$  of profit at least  $(1 - \epsilon)OPT$ . Note that all remaining squares have size at most  $\rho_\tau \leq (\epsilon/4)^{5+2^{\eta+1}}$ , as  $\tau \geq 1$ . Furthermore, since  $\epsilon \leq 1/3$ , then  $\eta \geq \log_{2/3}(1/3) > 2.7$ , and so

$$\rho_\tau = \rho_{\tau-1}^4 (\epsilon/4)^{5+2^{\eta+1}} \leq \rho_{\tau-1}^4 (\epsilon/4)^{5+2^{3.7}} < \epsilon/10^{10}, \tag{1}$$

which is very small compared to the bin. Hence, we can use NFDS [2] to re-pack small squares of profit at least  $(1 - 2\epsilon)OPT$  in the bin.

### 4 Instances with High Profit Big Squares

We now consider the case when every big square has profit larger than  $\epsilon OPT$ . Note that in this case,  $|\mathcal{B}^*| < 1/\epsilon$ .

#### 4.1 Blocks

We temporarily remove from  $P^*$  all medium and small squares. The empty space created is partitioned by the big squares into a set of disconnected regions  $r_1, r_2, \dots, r_\kappa$ . Each  $r_i$  is a polygonal region that might contain holes and it is delimited by a set  $p_i$  of polygonal boundaries. Let  $N_i$  be the number of vertices of  $p_i$ .

We divide each  $r_i$  into rectangular sub-regions by tracing a horizontal cutting line passing through each horizontal side of  $p_i$  and a vertical cutting line through each vertical side of  $p_i$ . These cutting lines define a grid of size at most  $N_i/2 \times N_i/2$  that splits  $r_i$  into  $O(N_i^2)$  rectangular regions called  $s_{j,k}$  (see Fig. 1).

Now we add back the medium and small squares in the same positions where they appear in the optimum solution  $P^*$ . Note that some of these squares might cross the block boundaries; we wish to simplify the packing so no square crosses

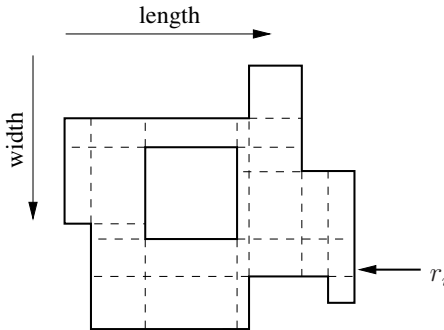


Fig. 1. Grid splitting region  $r_i$  into blocks

any block boundaries. To achieve this, some squares might need to be re-arranged and/or discarded. The exact procedure for eliminating intersections between squares and block boundaries depends on the dimensions and shape of each block. Accordingly, we partition the blocks in three classes as follows. For each block  $b_j$  let  $\widehat{Q}_j$  be the largest square whose interior intersects  $b_j$  and let  $\hat{s}_j$  be the size of  $\widehat{Q}_j$ . Note that  $\widehat{Q}_j \in \mathcal{M}^* \cup \mathcal{S}^*$ . Let

$$\alpha = 10/\epsilon \text{ and } \beta = 24/\epsilon^3. \tag{2}$$

**Definition 1**

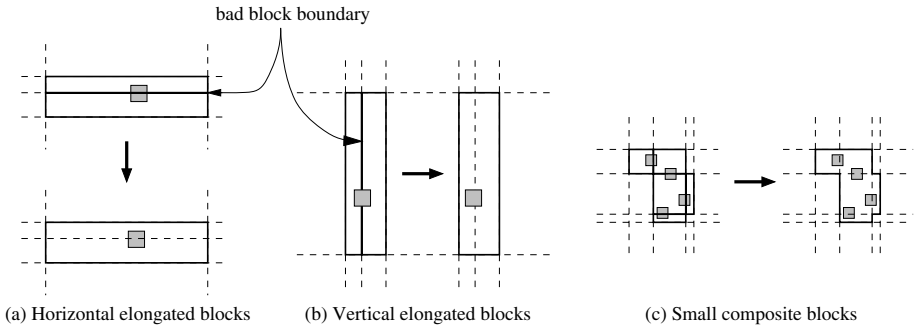
- $b_j$  **large**  $\alpha\hat{s}_j$
- $b_j$  **elongated**  $\beta\hat{s}_j$
- $\alpha\hat{s}_j$  **horizontal elongated block**
- $\alpha\hat{s}_j$   $\beta\hat{s}_j$  **vertical elongated block**
- $b_j$  **small**

To modify  $P^*$  so that no square crosses any block boundaries, first we need to eliminate all  $\dots$  (see Fig. 2). The common boundary  $I$  between two blocks  $b_i$  and  $b_j$  is  $\dots$  if there is some square  $Q_r$  crossing  $I$  and either

1.  $I$  is common to the long sides of two elongated blocks, or
2.  $I$  is common to a small block and to a long side of an elongated block, or
3.  $I$  is common to two small blocks.

Intuitively, bad block boundaries might be problematic because if a square  $Q_r$  crosses the long side of a small or elongated block  $b_j$ ,  $Q_r$  cannot be packed in  $b_j$  if it is larger than the smaller dimension of  $b_j$ .

We eliminate bad block boundaries by merging some of the blocks as follows. Consider one by one the columns of the grid splitting  $r_i$  (see Fig. 1). If some column has two adjacent horizontal elongated blocks sharing a bad boundary, these blocks are merged (see Fig. 2(a)). The resulting block is either horizontal elongated or large. This process eliminates all bad boundaries between horizontal



**Fig. 2.** Merging blocks

elongated blocks. Next, we consider one by one the rows of the grid. If in some row two adjacent vertical elongated blocks share a bad boundary, they are merged (see Fig. 2(b)). The resulting block is either vertical elongated or large.

After merging blocks as described, the only bad boundaries remaining might be between two small blocks, or between a small and an elongated block. If a group of blocks share bad boundaries, they are all merged into a composite small (see Fig. 2(c)). For convenience, a small block is also composite small.

**Lemma 1.**  $\dots r_i \dots N_i^2/4 \dots b_j \dots \alpha\beta(\hat{s}_j N_i/2)^2 \dots \hat{s}_j \dots b_j \dots N_i^2/4 \dots$

### 4.2 Assigning Squares to Blocks

Consider a region  $r_i$  and let  $C_i$  be the set of squares that cross the boundaries of  $r_i$ 's blocks. Each square completely contained in a block is assigned to that block. As for the squares in  $C_i$ , we assign them to blocks as follows.

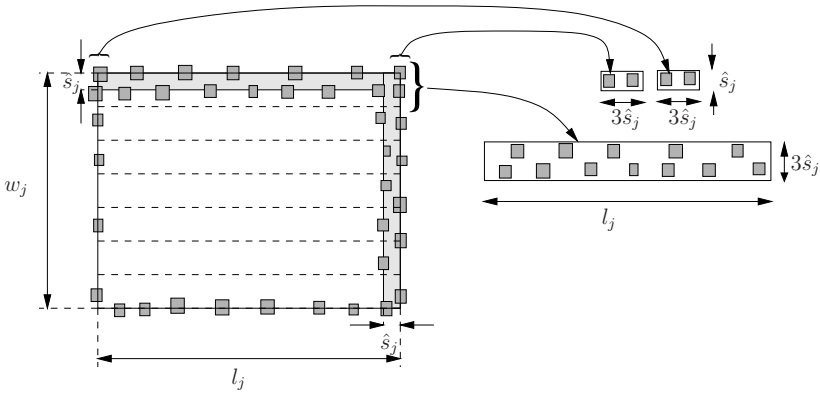
1. A square in  $C_i$  intersecting a large block  $b_j$  is assigned to  $b_j$ ; if a square intersects several large blocks, it is assigned to any one of them.
2. For the remaining squares in  $C_i$ , if a square  $Q_r$  intersects several elongated blocks, it is assigned to a block that intersects  $Q_r$  with its short side only; we can guarantee that this block exists because of the way in which blocks have been merged.

Note that every square in  $C_i$  is allocated to a large or elongated block, as no square crosses the common boundary of two small blocks (otherwise they would have been merged). We now modify  $P^*$  so that no square crosses any block boundaries.

### 4.3 Large Blocks

Consider a large block  $b_j$ . We only consider the case when the width  $w_j$  of  $b_j$  is smaller than its length  $l_j$ . Let  $S_j$  be the set of squares assigned to  $b_j$  and let  $\hat{Q}_j$  be the largest square in  $S_j$ . The following shifting technique allows us to (i) round the dimensions of  $b_j$  down to the nearest multiple of  $\hat{s}_j$  (the size of  $\hat{Q}_j$ ) and (ii) also allows us to remove the squares crossing the boundaries of  $b_j$ , while losing only a very small fraction of the total profit of the squares originally packed by  $P^*$  in  $b_j$ .

Let  $D_j$  be the set of squares crossing any of the boundaries of block  $b_j$ , plus the squares whose top or right sides are at distance smaller than  $\hat{s}_j$  from the top or the right side of  $b_j$  (see Fig. 3).  $D_j$  is momentarily removed from the packing. This creates an empty region of width at least  $\hat{s}_j$  at the top of  $b_j$  and an empty region of length at least  $\hat{s}_j$  at the right side of  $b_j$ . These empty regions allow us to round the width and length of  $b_j$  down to the nearest multiple of  $\hat{s}_j$  without having to remove any more squares from the packing.



**Fig. 3.** The shifting technique. Set  $D_j$  appears in dark shade.

Split  $b_j$  into  $\alpha/10$  horizontal strips of the same width, as shown in Fig. 3. Each square  $Q_k$  packed inside  $b_j$  is assigned to one of the strips as follows: If  $Q_k$  intersects only one strip  $t$ , then  $Q_k$  is assigned to  $t$ ; if  $Q_k$  intersects two strips  $t$  and  $t'$ , then  $Q_k$  is assigned to the strip on the top. Since every strip has width at least  $w_j/(\alpha/10) \geq 10\hat{s}_j$ , by Def. 1, then, no square intersects three strips.

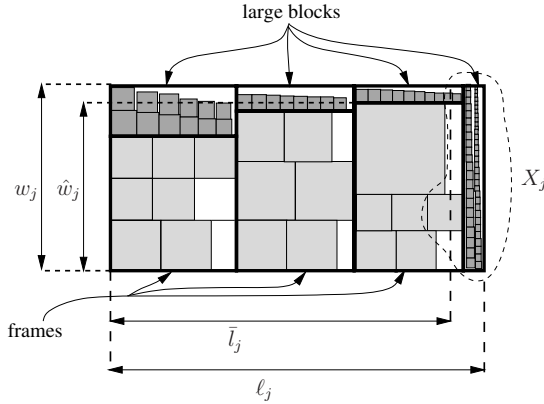
Let  $t_j$  be the strip for which the total profit,  $\bullet_{\gamma} \dots (t_j)$ , of the squares assigned to it is minimum. Clearly,  $\bullet_{\gamma} \dots (t_j) \leq \frac{10}{\alpha} \bullet_{\gamma} \dots (S_j) \leq \epsilon \times \bullet_{\gamma} \dots (S_j)$ , by (2). Hence, if we remove from  $P^*$  all squares assigned to  $t_j$  the profit loss will be very small. Furthermore, this creates inside  $t_j$  an empty region of width at least  $9\hat{s}_j$ . We use this empty space to reintroduce the previously removed squares  $D_j$ :

- All squares in  $D_j$  that were taken off the top of  $b_j$  can be packed in a strip of length  $l_j + 2\hat{s}_j$  and width  $3\hat{s}_j$  (see Fig. 3). These squares can be re-arranged so they fit in two strips: one of length  $l_j$  and width  $3\hat{s}_j$  and the other of length  $6\hat{s}_j$  and width  $\hat{s}_j$  (see right hand side of Fig. 3).
- All remaining squares in  $D_j$  that were removed from the right side of  $b_j$  can be packed in a strip of length  $3\hat{s}_j$  and width  $w_j < l_j$  (see Fig. 3).
- The other squares that crossed the bottom of  $b_j$  can be packed in a strip of length  $l_j$  and width  $\hat{s}_j$ , while the squares that crossed the left side of  $b_j$  can be packed in a strip of length  $w_j < l_j$  and width  $\hat{s}_j$ .

The total profit of the squares that remain in  $b_j$  is at least  $(1 - \epsilon) \bullet_{\gamma} \dots (S_j)$ . Since every square packed in  $b_j$  has size at most  $\hat{s}_j$  (which is very small compared to the dimensions of  $b_j$ ), we can use the NFDS algorithm to produce a simple-structured packing for  $b_j$ . The total profit of the squares that the NFDS algorithm can pack in  $b_j$  is at least  $(1 - 2\epsilon) \bullet_{\gamma} \dots (S_j)$ .

#### 4.4 Elongated Blocks

To understand how we will modify the structure of the packing for elongated blocks, we first give a brief review of the strip packing algorithm of Kenyon and Rémila [8], which we will refer to as the KR algorithm.



**Fig. 4.** Packing produced by the KR algorithm. Set  $S_{\epsilon j}$  appears in light shading.

**4.4.1 The KR Algorithm**

The KR algorithm uses a two-stage process to pack a set  $S_j$  of squares in an elongated block  $b_j$ . For convenience, we assume that the length  $l_j$  of  $b_j$  is larger than its width  $w_j$ . Let  $\widehat{Q}_j$  be the largest square packed in  $b_j$  and let  $\hat{s}_j$  be its size. In the first stage, the KR algorithm packs in  $b_j$  the set  $S_{\epsilon j} \subseteq S_j$  of squares of size at least  $\epsilon \times w_j / (2 + \epsilon)$  using a linear programming approach. To do this, a (multi)subset  $T_j \subseteq S_{\epsilon j}$  of  $h = \left(\frac{2+\epsilon}{\epsilon}\right)^2$  squares is selected; the width of each square in  $S_{\epsilon j}$  is rounded up to the width of the nearest square from  $T_j$ . Rounded squares are grouped into  $h$  clusters,  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_h$ , called  $\dots$ . Configuration  $\mathcal{C}_i$  has width  $W_i \leq w_j$ . The solution for the linear program determines the length  $L_i$  of each  $\mathcal{C}_i$ .

Each configuration  $\mathcal{C}_i$  is allocated a rectangular region  $R_i$  of width  $W_i$  and length  $L_i$ . The rectangular regions  $R_1, R_2, \dots, R_h$  are called  $\dots$ . If configuration  $\mathcal{C}_i$  is composed of squares of widths  $w_{i1}, w_{i2}, \dots, w_{ir}$ , then frame  $R_i$  is divided into  $r$  rows of widths  $w_{i1}, w_{i2}, \dots, w_{ir}$ . Rectangles from  $S_{\epsilon j}$  of width  $w_{ij}$  are packed in the row of width  $w_{ij}$  as long as their total length does not exceed  $L_i + \hat{s}_j$  (see Fig. 4).

In the second stage the space remaining in  $b_j$  is divided into  $h + 1$  rectangular blocks,  $R'_1, R'_2, \dots, R'_h, R'_{h+1}$ , that for convenience we call large blocks;  $R'_i, 1 \leq i \leq h$ , has width  $w_j - W_i$  and length  $L_i$ . Block  $R'_{h+1}$  has width  $w_j$  and length  $L_j - \sum_{i=1}^h L_i$ . Squares in  $S_j \setminus S_{\epsilon j}$  are packed in these blocks with the NFDS algorithm. The packing produced has width no larger than  $w_j$  and length [8]

$$l_j \leq l_j(1 + \epsilon) + (4(2 + \epsilon)^2/\epsilon^2 + 1)\hat{s}_j < l_j(1 + \epsilon) + 22\hat{s}_j/\epsilon^2, \text{ as } \epsilon \leq 1/3. \quad (3)$$

**4.4.2 Modifying the Packing in an Elongated Block**

Consider an elongated block  $b_j$ . We only look at the case when the length  $l_j$  of  $b_j$  is larger than its width  $w_j$ . Let  $S_j$  be the set of squares assigned to  $b_j$  and let

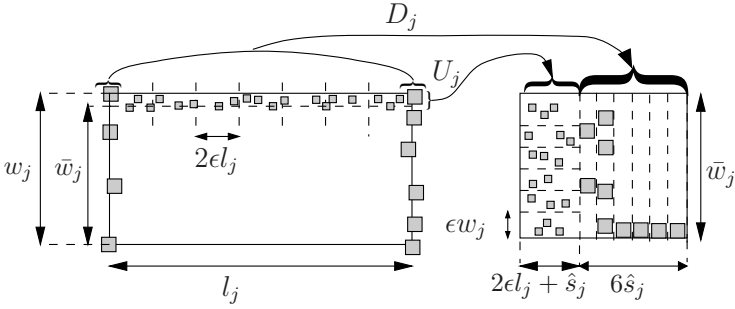


Fig. 5. Re-packing the squares  $D_j \cup U_j$

$\hat{s}_j$  be the size of the largest square in  $S_j$ . All squares in  $S_j$  have size at most  $w_j$  as they are either completely contained inside  $b_j$  or they intersect the short side of  $b_j$ .

Let  $D_j$  be the set of squares crossing the short sides of  $b_j$ . These squares are, for the time being, removed from  $b_j$ . Let  $S'_j$  be the squares remaining in  $b_j$ . We re-pack  $S'_j$  by using the KR algorithm. In this new packing, let  $\hat{w}_j$  be the width of the widest configuration used (see Fig. 4). Let  $i$  be the largest integer such that  $\hat{w}_j + i\epsilon\hat{s}_j \leq w_j$ . We round the width of  $b_j$  down to  $\bar{w}_j = \max\{\hat{s}_j, \hat{w}_j + i\epsilon\hat{s}_j\}$ . By doing this, some subset  $U_j \subseteq S'_j$  of squares of size smaller than  $\epsilon w_j / (2 + \epsilon)$  might not fit at the top of  $b_j$  (see Fig. 4; the dark shaded squares have size smaller than  $\epsilon w_j / (2 + \epsilon)$ ). The total area of the squares in  $U_j$  is smaller than  $2\epsilon w_j l_j / (2 + \epsilon)$ . The squares  $U_j$  are removed from  $b_j$ ; they will be re-introduced to the packing later.

Now, we round the length of  $b_j$  down to the nearest multiple  $\bar{l}_j$  of  $\hat{s}_j$  (see Fig. 4). The KR algorithm might produce a packing for  $S'_j$  of length larger than  $\bar{l}_j$ , but we can reduce the length to at most  $\bar{l}_j$  by using the shifting technique described in Section 4.3. Let  $X_j$  be the set of squares that are not completely packed by the KR algorithm inside the rounded block  $b_j$  (see Fig. 4). We divide  $b_j$  into  $1/(4\epsilon)$  strips of the same length and width  $\bar{w}_j$ . For at least one of these strips  $t_j$ , the total profit of the squares completely packed in it is at most  $4\epsilon \times \dots (S'_j)$ . Remove these squares from  $t_j$  creating an empty region of width  $\bar{w}_j$  and length at least  $4\epsilon\bar{l}_j - 2\hat{s}_j \geq 3\epsilon\bar{l}_j + \epsilon\beta\hat{s}_j - 2\hat{s}_j$ , since by Def. 1,  $l_j \geq \beta\hat{s}_j$ .

By (3),  $X_j$  can be packed in a rectangle of width  $\bar{w}_j$  and length  $\epsilon\bar{l}_j + 22\hat{s}_j/\epsilon^2 + \hat{s}_j$ . We can pack  $X_j$  in  $t_j$  and still have an empty region of width  $\bar{w}_j$  and length

$$\ell \geq 3\epsilon\bar{l}_j + \epsilon\beta\hat{s}_j - 2\hat{s}_j - (\epsilon\bar{l}_j + 22\hat{s}_j/\epsilon^2 + \hat{s}_j) = 2\epsilon\bar{l}_j + \hat{s}_j(\frac{2}{\epsilon^2} - 3), \text{ by (2), } \geq 2\epsilon\bar{l}_j + 15\hat{s}_j.$$

The squares in  $D_j \cup U_j$  can be added back and packed inside  $t_j$  in a region of width  $\bar{w}_j$  and length  $2\epsilon\bar{l}_j + 5\hat{s}_j < \ell$ . To see this, observe that the squares from  $D_j$  can be packed in two strips of width  $w_j + 2\hat{s}_j$  and length  $\hat{s}_j$  (see Fig. 5). Since  $\bar{w}_j \geq w_j - \epsilon\hat{s}_j$  and  $\bar{w}_j \geq \hat{s}_j$ , then  $D_j$  can also be packed in six strips of width  $\bar{w}_j$  and length  $\hat{s}_j$ . The squares in  $U_j$  can be packed in a horizontal strip

of length  $l_j$  and width  $2\epsilon w_j/(2 + \epsilon) < \epsilon w_j$ . These squares can be re-arranged so they fit in at most  $1/(2\epsilon)$  strips of length  $2\epsilon l_j + \hat{s}_j$  and width  $\epsilon w_j$ . The total width of these strips is at most  $\epsilon w_j/(2\epsilon) = w_j/2 < w_j - \hat{s}_j/2 < w_j - \epsilon \hat{s}_j \leq \bar{w}_j$ , as  $\hat{s}_j \leq w_j$  and  $\epsilon \leq 1/3$ .

The above process packs in the rounded block  $b_j$  all the squares of  $S_j$ , except those originally placed in strip  $t_j$ . The total profit of the packed squares is at least  $(1 - 4\epsilon) \bullet \dots \bullet (S_j)$ , but now no square crosses the boundaries of  $b_j$  and, furthermore, the squares are packed in a very regular manner.

### 4.5 Composite Small Blocks

Let  $B_1$  be the set of composite small blocks and let  $S_1$  be the set of squares packed in the blocks of  $B_1$ . Recall that we are considering the case when every big square has profit larger than  $\epsilon OPT$ . Since  $S_1$  contains only medium and small squares, then,  $\bullet \dots \bullet (S_1) \leq (1 - \epsilon)OPT$ .

Let  $b_{j1} \in B_1$  be a composite small block and let  $S_{j1} \subseteq S_1$  be the set of squares packed in  $b_{j1}$ . We will modify the way in which  $S_{j1}$  is packed inside  $b_{j1}$  by recursively applying to  $b_{j1}$  the transformations described in Sections 4, 5, i.e., we will consider that now  $b_{j1}$  is the bin  $\mathcal{R}$  where the squares need to be packed (however, now the bin might not be rectangular). Accordingly, we define  $\rho'_0 = 1$ ,  $\rho'_k = (\rho'_{k-1})^4(\epsilon/4)^{5+2^{n+1}}$ , and  $\rho_k = \rho'_k \times \hat{s}_{j1}$ , for all integers  $k \geq 1$ , where  $\hat{s}_{j1}$  is the size of the largest square in  $S_{j1}$ . Then, we define the sets  $\mathcal{Q}_{kj}^* = \{Q_i \in S_{j1} \mid s_i \in (\rho_k, \rho_{k-1}]\}$ ,  $k \geq 1$ . Let  $\tau \geq 2$  be the smallest index for which  $\bullet \dots \bullet (\mathcal{Q}_{\tau j}^*) \leq \epsilon \times \bullet \dots \bullet (S_{j1})$ . The set  $S_{j1}$  is then partitioned into big  $\mathcal{B}_{j1}^* = \{Q_i \in S_{j1} \mid s_i > \rho_{\tau-1}\}$ , medium  $\mathcal{M}_{j1}^* = \{Q_i \in S_{j1} \mid \rho_{\tau} < s_i \leq \rho_{\tau-1}\}$ , and small  $\mathcal{S}_{j1}^* = \{Q_i \in S_{j1} \mid s_i \leq \rho_{\tau}\}$  squares as we did in Section 2. For simplicity, we required  $\tau \geq 2$  to ensure that the set  $\mathcal{B}_{j1}^*$  is not empty. We now need to consider two cases:

- (i) The set  $\mathcal{B}_{j1}^*$  contains a square  $Q_\ell$  of profit no larger than  $\epsilon \times \bullet \dots \bullet (S_{j1})$ . In this case we modify the packing for  $S_{j1}$  as indicated in Section 5.
- (ii) Every square in  $\mathcal{B}_{j1}^*$  has profit larger than  $\bullet \dots \bullet (S_{j1})$ . We recursively partition  $b_{j1}$  into smaller blocks using the procedure described in Section 4.

With each recursive partitioning, the composite small blocks might get more complex shapes, since composite small blocks are constructed by merging several rectangular regions together. Fortunately, we only need to recursively partition these blocks a constant number of times, as we show below.

Let  $B_2$  be the set of composite small blocks created after recursively partitioning all the blocks in  $B_1$ . Since in each block  $b_{j1} \in B_1$  that needs to be partitioned there must be at least one square  $Q \in \mathcal{B}_{j1}^*$  of profit larger than  $\epsilon \times \bullet \dots \bullet (S_{j1})$ , then the total profit of the squares packed in the smaller blocks created after partitioning  $b_{j1}$  is at most  $(1 - \epsilon) \bullet \dots \bullet (S_{j1})$ . Therefore, the total profit of the set  $S_2$  of squares packed in all the blocks of  $B_2$  is  $\bullet \dots \bullet (S_2) \leq (1 - \epsilon) \sum_{b_{j1} \in B_1} \bullet \dots \bullet (S_{j1}) = (1 - \epsilon) \bullet \dots \bullet (S_1) \leq (1 - \epsilon)^2 OPT$ .

Let  $B_i$  be the set of composite small blocks after  $i$  recursive applications of the partitioning procedure and let  $S_i$  be the set of squares packed in  $B_i$ . By the

above discussion, by recursively partitioning the blocks in  $B_i$  we create a new set of composite small blocks  $B_{i+1}$  such that the total profit of the set  $S_{i+1}$  of squares packed in  $B_{i+1}$  is  $\bullet_{\gamma} \dots(S_{i+1}) \leq (1 - \epsilon)\bullet_{\gamma} \dots(S_i) \leq (1 - \epsilon)^i OPT$ .

If we perform this recursive partitioning  $\eta = \log_{1-\epsilon}(\epsilon)$  times, the total profit of the squares packed inside the blocks  $B_\eta$  is at most  $(1 - \epsilon)^\eta OPT = \epsilon \times OPT$ . Thus, we simply discard all squares packed by the optimum solution  $P^*$  in the blocks  $B_\eta$ , losing only profit at most  $\epsilon \times OPT$ . We can show that

$$|B_\eta| < \left(\frac{4}{\epsilon}\right)^{2^{\eta+2}} \quad \text{and} \quad n_\eta < \left(\frac{4}{\epsilon}\right)^{2^{\eta+1}}, \tag{4}$$

where  $n_\eta$  is the maximum number of vertices in any polygonal region delimiting a composite small block (we omit the proofs, due to space limitations).

This recursive partitioning creates a packing with a hierarchical structure for the composite small blocks. Large squares relative to the size of a composite small block split it into large, elongated, and small sub-blocks where squares are packed as indicated in this section. As we show below, the number and shape of the composite small blocks affects the time complexity of our algorithm.

### 5 Instances with a Low Profit Big Square

Let  $b_j$  be either the unit size square bin, or one of the composite small blocks created during the recursive partitioning procedure described above. Let  $S_j$  be the set of squares packed in  $b_j$  by the optimum solution  $P^*$ . We partition  $S_j$  into 3 sets:  $\mathcal{B}_j^*$ ,  $\mathcal{M}_j^*$ , and  $\mathcal{S}_j^*$ , of big, medium, and small squares as described in Section 4.5. If  $b_j$  is a composite small block then,  $b_j$  was obtained by merging at most  $n_\eta$  rectangular blocks, each of width and length smaller than  $\alpha\hat{s}_j$ , where  $\hat{s}_j$  is the size of the largest square in  $S_j$ . So, the total area of  $b_j$  is smaller than  $n_\eta(\alpha\hat{s}_j)^2$  and the (constant) number  $n_j^*$  of big squares in  $\mathcal{B}_j^*$  is

$$n_j^* < n_\eta(\alpha\hat{s}_j)^2 / \rho_{\tau-1}^2 < n_\eta\alpha^2 / \rho_{\tau-1}^2 = n_\eta\alpha^2(4/\epsilon)^{2(5+2^{\eta+1})(4^{1/\epsilon}-1)/3}. \tag{5}$$

Let us remove the medium squares  $\mathcal{M}_j^*$  from  $b_j$ ; causing a profit loss of value at most  $\epsilon \times \bullet_{\gamma} \dots(S_j)$ . This creates a size gap between the big and small squares. We also remove the small squares from  $b_j$ , but we will add them back later.

Let  $Q_{\min} \in \mathcal{B}_j^*$  be a big square of profit at most  $\epsilon \times \bullet_{\gamma} \dots(S_j)$  and  $p_j$  be the polygon delimiting  $b_j$ . The empty space in  $b_j$  left by the  $n_j^*$  big squares can be split with horizontal cutting lines starting at the vertices of  $p_j$  and at the vertices of the squares in  $\mathcal{B}_j^*$  into  $m$  rectangles  $\mathcal{R}_j = \{R_1, R_2, \dots, R_m\}$ . For convenience, we call these rectangles, large blocks. By (4),  $p_j$  has at most  $n_\eta$  vertices and since the total number of vertices of the big squares in  $\mathcal{B}_j^*$  is  $4n_j^*$ , then,

$$m < n_\eta + 4n_j^*. \tag{6}$$

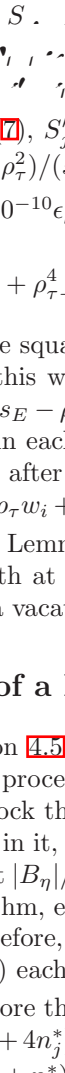
Now, add back the small squares and, then, round the width and length of each large block  $R_i$  down to the nearest value of the form  $k\hat{s}$ , where  $k$  is an



integer value and  $\hat{s} \leq \rho_\tau$  is the size of the largest small square in  $\mathcal{S}_j^*$ . After doing this, a set  $S'_j$  of small squares might not completely fit in the blocks  $\mathcal{R}_j$ . Let  $l_i$  be the length of block  $R_i$  and  $w_i$  be its width. We can show that the total area of the squares in  $S'_j$  is

$$A'_j < \frac{1}{5}\rho_{\tau-1}^2 \tag{7}$$

Let us remove  $Q_{\min}$  from  $b_j$ , losing additional profit of value at most  $\epsilon OPT$ . This creates an empty square region  $E$  in  $b_j$  of area at least  $\rho_{\tau-1} \times \rho_{\tau-1}$  where we can pack the small squares  $S'_j$  by using the NFDS algorithm. Let  $s'$  be the size of the largest square in  $S'_j$  and let  $s_E$  be the size of any side of  $E$ . Round  $s_E$  down to the nearest multiple of  $s'$ . Note that after rounding,  $s_E > \rho_{\tau-1} - s' \geq \rho_{\tau-1} - \rho_\tau$ .

**Lemma 2.** □   $w \leq (AREA(S) + \delta \ell - \delta^2) / (\ell - \delta)$

By Lemma 2 and (7),  $S'_j$  can be packed in a bin of length  $s_E$  and width  $w'_j \leq (\rho_{\tau-1}^2/5 + \rho_\tau s_E - \rho_\tau^2) / (s_E - \rho_\tau) < (\rho_{\tau-1}^2/5 + \rho_\tau) / (\rho_{\tau-1} - 2\rho_\tau)$ . Since  $\rho_\tau = \rho_{\tau-1}^4 (\frac{\epsilon}{4})^{5+2^{\eta+1}} < 10^{-10} \epsilon \rho_{\tau-1}^4$ , by (1),  $< 10^{-10} \rho_{\tau-1}$ , then

$$w'_j < (\rho_{\tau-1}^2/5 + \rho_{\tau-1}^4 (\frac{\epsilon}{4})^{5+2^{\eta+1}}) / (\rho_{\tau-1} - 2 \times 10^{-10} \rho_{\tau-1}) < \rho_{\tau-1}/4. \tag{8}$$

Therefore, all the squares in  $S'_j$  fit in the empty space left by  $Q_{\min}$ . Furthermore, after doing this we still have left an empty region of length at least  $s_E$  and width at least  $s_E - \rho_{\tau-1}/4 \geq \frac{3}{4}\rho_{\tau-1} - \rho_\tau$ . We now use NFDS to repack the squares contained in each block  $R_i \in \mathcal{R}_j^*$ . Some of the small squares might not fit in the blocks  $R_i$  after being re-arranged by NFDS, but their total area [2] is at most  $\sum_{R_i \in \mathcal{R}_j^*} (2\rho_\tau w_i + \rho_\tau l_i) < A'_j < \rho_{\tau-1}^2/5$ , where  $w_i$  is the width of  $R_i$  and  $l_i$  is its length. By Lemma 2 and (8), these squares can be packed in a bin of length  $s_E$  and width at most  $\rho_{\tau-1}/4$ . Hence, they fit in the empty space that remains in the area vacated by  $Q_{\min}$ .

## 6 Structure of a Near Optimum Solution

As shown in Section 4.5, the total number of composite small blocks that our recursive splitting procedure creates is at most  $|B_\eta| < (4/\epsilon)^{2^{\eta+2}}$ . Since each composite small block that needs to be recursively split must have at most  $1/\epsilon$  big squares packed in it, the number of big squares involved in these  $\eta$  recursive splittings is at most  $|B_\eta|/\epsilon$ . In the final set  $\tilde{S}$  of blocks produced by the recursive partitioning algorithm, each one of these blocks could contain a large square of low profit, and therefore, these blocks might need to be further split as described in Section 5. By (5) each block  $b_j \in \tilde{S}$  could store  $n_j^*$  big squares, which by (6), divide  $b_j$  into no more than  $n_\eta + 4n_j^*$  blocks. Therefore,  $\tilde{S}$  could be further split into  $N_B \leq |B_\eta|(n_\eta + 4n_j^*)$  blocks. The total number of big squares packed in the bin is  $N_S \leq |B_\eta|(\frac{1}{\epsilon} + n_j^*)$ .

The transformations described in Sections 2.4-2.5 prove that for any instance  $Q$  of the square packing problem there is a near optimum solution  $P^+$  that selects a subset  $Q^+$  of squares of total profit  $(1 - O(\epsilon))OPT$  and packs them in the unit size square bin in a very regular manner:

- The packing  $P^+$  labels a constant number  $N_S \leq |B_\eta|(n_j^* + 1/\epsilon)$  of squares as  $s_1, \dots, s_{N_S}$ .
- The space not occupied by the big squares can be partitioned into a constant number  $N_B \leq |B_\eta|(n_\eta + 4n_j^*)$  of large and elongated rectangular blocks.
- The remaining, unlabelled, squares are packed in the blocks so that none crosses any block boundaries.
- A large block  $b_i$  has length  $l_i$  and width  $w_i$  that are multiples of the size  $\hat{s}_i$  of the largest square packed in  $b_i$ . Squares of total area at most  $(l_i - 2\hat{s}_i)(w_i - \hat{s}_i)$  are packed in  $b_i$  using the NFDS algorithm.
- An elongated block  $b_j$  has one of its dimensions much larger (by at least a factor  $\beta/\alpha$ ) than the other one. The larger dimension of  $b_j$  is a multiple of the size  $\hat{s}_j$  of the largest square packed in  $b_j$ ; the smaller dimension is a multiple of  $\epsilon \times \hat{s}_j$ . Squares are packed in  $b_j$  using the KR algorithm.

## 7 The Algorithm

Our approximation algorithm for the square packing problem is relatively simple, as it just enumerates a polynomial number of packings with the structure described above and then it selects a packing with the highest profit. Some care is needed when selecting the squares to pack in each block to ensure that the solution produced has profit close to that of  $P^*$ .

Let  $n_L$  and  $n_E$  be the number of large and elongated blocks in  $P^+$ , respectively. The algorithm tries all non-negative values for  $n_L$  and  $n_E$  such that  $n_L + n_E \leq N_B$ . For every choice of  $n_L$  and  $n_E$  the algorithm must select the dimensions of each long and elongated block. To determine the dimensions of a block  $b_i$ , we first choose a square  $\hat{Q}_i \in Q$  to be the largest square packed in that block. If  $b_i$  is a large block, then its width  $w_i$  and length  $l_i$  are multiples of the size  $\hat{s}_i$  of  $\hat{Q}_i$ . Note that  $w_i$  and  $l_i$  are at most  $n\hat{s}_i$ , so there are  $O(n^2)$  choices for the dimensions of  $b_i$ . If  $b_i$  is an elongated block, then its smaller dimension is a multiple of  $\epsilon\hat{s}_i$  and its larger dimension is a multiple of  $\hat{s}_i$ . Hence, there are up to  $n^2/\epsilon$  possible choices for the dimensions of  $b_i$  in this case.

For each choice of the number of blocks and the dimensions of each block, we need to select the set  $\mathcal{G}$  of big squares to be packed in the bin. Since  $|\mathcal{G}| \leq N_S$ , the algorithm tries for  $\mathcal{G}$  all subsets of  $Q$  of up to  $N_S$  squares. The number of possible subsets is  $O(n^{N_S})$ , which is polynomial in  $n$  as  $N_S$  is constant.

Let  $\mathcal{L}$  be the set of blocks selected by the algorithm. The algorithm takes this set  $\mathcal{L} \cup \mathcal{G}$  of rectangles and tries to pack them in the unit size bin  $\mathcal{R}$ . Since  $|\mathcal{L} \cup \mathcal{G}|$  is constant, we can in  $O(1)$  time either find a packing for  $\mathcal{L} \cup \mathcal{G}$  or decide that no packing for them exists. To do this, let us first define an  $(\epsilon, \beta)$ -packing for a set  $S$  of rectangles as a packing in which no rectangle can be shifted

up or to the left without overlapping other rectangles or crossing the boundaries of the bin. Clearly, any packing for  $S$  can be transformed into an upper-left justified packing by repeatedly shifting the rectangles up and/or to the left until no rectangle can be further moved. In an upper-left justified packing for  $S$ , every square  $Q_j \in S$  has its upper left corner at a distance  $x_j$  (which is the sum of lengths of at most  $|S|$  rectangles) from the left side of the bin and at a distance  $y_j$  (which is the sum of widths of at most  $|S|$  rectangles) from the top of the bin. Therefore, for each square  $Q_j$  there are at most  $2^{|S|}$  possible values for  $x_j$  and  $2^{|S|}$  possible values for  $y_j$ .

To pack  $\mathcal{L} \cup \mathcal{G}$ , we can simply compute for each rectangle  $r \in \mathcal{L} \cup \mathcal{G}$  the at most  $2^{|\mathcal{L} \cup \mathcal{G}|} \times 2^{|\mathcal{L} \cup \mathcal{G}|}$  possible coordinates for its upper-left corner. If for one of these positionings all the rectangles fit in the bin without overlapping, then we have found a valid packing for them, otherwise no packing exists.

For each set  $\mathcal{L}$  of blocks and  $\mathcal{G}$  of big squares that can be packed in the bin, we consider the set  $\mathcal{E} \subseteq \mathcal{L}$  of elongated blocks. Each elongated block  $b_j \in \mathcal{E}$  needs to be split into frames and large blocks as described in Section 4.4. To show how this splitting is done, let us consider a block  $b_i \in \mathcal{E}$ . For simplicity we assume that  $w_i < l_i$ . First, we select a (multi)subset  $T_i$  of  $h = \left(\frac{2+\epsilon}{\epsilon}\right)^2$  squares that will be used to split  $b_i$  into  $h$  frames and  $h + 1$  large blocks as explained in Section 4.4.1. The length of each frame is a multiple of  $\hat{s}_j$  and its width is the sum of widths of a subset of squares from  $T_j$ ; therefore, for each frame there are at most  $n2^h$  possible values for its dimensions. Since there are  $h$  frames and  $n^h$  possible choices for  $T_j$ , there are  $(n2^h)^h n^h$  ways of selecting frames for block  $b_j$ . For each choice of frames there is only one way of splitting the remaining space of  $b_j$  into large blocks.

The above process eliminates the elongated blocks from  $\mathcal{L}$ , but adds a new set of large blocks. Let  $\mathcal{L}$  be the resulting set of large blocks and  $\mathcal{F}$  be the set of frames created by splitting the elongated blocks. The final number  $N_L$  of large blocks is then  $N_L = N_B + |\mathcal{E}|(h + 1)$  and the number  $N_F$  of frames is  $N_F = |\mathcal{E}|h$ .

### 7.1 Selecting the Squares

The final step is to allocate a set of squares to each block and frame, and to pack them there. For each block  $b_j \in \mathcal{L}$ , of length  $l_j$  and width  $w_j$ , we select a square  $\hat{Q}_j \in Q$ , of size  $\hat{s}_j$ , to be the largest square packed in it. We pack in  $b_j$  only squares of size at most  $\hat{s}_j$ , and total area at most  $(l_j - 2\hat{s}_j)(w_j - \hat{s}_j)$ . For each frame  $f_j \in \mathcal{F}$  we choose two squares  $\check{Q}_j$  and  $\hat{Q}_j$  of sizes  $\check{s}_j$  and  $\hat{s}_j$ ,  $\check{s}_j \leq \hat{s}_j$ . We can pack in  $f_j$  only squares of size at least  $\check{s}_j$  and at most  $\hat{s}_j$ . If the total length  $l_j$  of  $f_j$  is smaller than its width, then the length of each square allocated to  $f_j$  is rounded up to  $\hat{s}_j$ , otherwise, the width of each square to be packed in  $f_j$  is rounded up to  $\hat{s}_j$ . The total area of the squares assigned to  $f_j$  must be at most  $w_j l_j$ .

Now, we must allocate to the blocks and frames  $\mathcal{L} \cup \mathcal{F}$  a maximum profit subset of squares that satisfy the above conditions. This allocation problem is a special instance of the generalized assignment problem [3]. However, since  $|\mathcal{L} \cup \mathcal{F}|$

is constant, a straightforward extension of the  $O(n^2/\epsilon)$  FPTAS of Lawler [9] for the knapsack problem can be used to make the allocation. The only change that we need to make to the algorithm in [9] is that instead of computing single pairs  $(\bullet, \dots(S), \dots(S))$  for candidate subsets  $S$  of squares, we need to compute  $N_L + N_B$  pairs  $(\bullet, \dots(S_1), \dots(S_1)), \dots, (\bullet, \dots(S_{N_L+N_B}), \dots(S_{N_L+N_B}))$  for those candidate sets indicating how the squares are allocated to the blocks and frames. This change increases the time complexity of the algorithm to  $O(n^2(N_L + N_B)/\epsilon)$ .

The total profit of the squares allocated by this algorithm is at least  $(1 - \epsilon)p^+$ , where  $p^+$  is the total profit of the squares allocated by  $P^+$  to the large blocks and frames. As discussed above, all squares allocated to a large block  $b_j$  are packed using the NFDS algorithm. Furthermore, all squares assigned to a frame  $f_j$  are packed by simply placing the squares side by side in the frame.

**Theorem 1.** *Let  $\epsilon \in (0, 1)$  and  $n \geq 1/\epsilon$ . Then there is an algorithm that runs in  $O(n^2(N_L + N_B)/\epsilon)$  time and packs a set of squares of total profit at least  $(1 - \epsilon)p^+$ .*

## References

1. Bansal, N., Sviridenko, M.: Two-dimensional bin packing with one dimensional resource augmentation. *Discrete Optimization* (to appear)
2. Coffman, E.G., Garey, M.R., Johnson, D.S., Tarjan, R.E.: Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing* 9, 808–826 (1980)
3. Cohen, R., Katzir, L., Raz, D.: An Efficient Approximation for the Generalized Assignment Problem. *Information Processing Letters* 100(4), 162–166
4. Fishkin, A.V., Gerber, O., Jansen, K., Solis-Oba, R.: Packing weighted rectangles into a square. In: Jdrzejowicz, J., Szepietowski, A. (eds.) *MFCS 2005*. LNCS, vol. 3618, pp. 352–363. Springer, Heidelberg (2005)
5. Harren, R.: Approximating the orthogonal knapsack problem for hypercubes. In: *International Colloquium on Automata Languages and Programming (ICALP 2006)*, pp. 238–249 (2006)
6. Jansen, K., Zhang, G.: Maximizing the total profit of rectangles packed into a rectangle. *Algorithmica* 47, 323–342 (2007)
7. Kleitman, D.J., Krieger, M.M.: An optimal bound for two dimensional bin packing. In: *Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1975)*, pp. 163–168 (1975)
8. Kenyon, C., Rémila, E.: A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research* 25, 645–656 (2000)
9. Lawler, E.: Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research* 4, 339–356 (1979)
10. Leung, J.Y.-T., Tam, T.W., Wong, C.S., Young, G.H., Chin, F.Y.L.: Packing squares into a square. *Journal of Parallel and Distributed Computing* 10, 271–275 (1990)
11. Novotny, P.: On packing of squares into a rectangle. *Archivum Mathematicum* 32, 75–83 (1996)

# Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints<sup>\*,\*\*</sup>

Juan Pablo Vielma and George L. Nemhauser

H. Milton Stewart School of Industrial and Systems Engineering,  
Georgia Institute of Technology, Atlanta, GA, USA  
{jvielma,gnemhaus}@isye.gatech.edu

**Abstract.** Many combinatorial constraints over continuous variables such as SOS1 and SOS2 constraints can be interpreted as disjunctive constraints that restrict the variables to lie in the union of  $m$  specially structured polyhedra. Known mixed integer binary formulations for these constraints have a number of binary variables and extra constraints that is linear in  $m$ . We give sufficient conditions for constructing formulations for these constraints with a number of binary variables and extra constraints that is logarithmic in  $m$ . Using these conditions we introduce the first mixed integer binary formulations for SOS1 and SOS2 constraints that use a number of binary variables and extra constraints that is logarithmic in the number of continuous variables. We also introduce the first mixed integer binary formulations for piecewise linear functions of one and two variables that use a number of binary variables and extra constraints that is logarithmic in the number of linear pieces of the functions. We prove that the new formulations for piecewise linear functions have favorable tightness properties and present computational results showing that they can significantly outperform other mixed integer binary formulations.

## 1 Introduction

An important question in the area of mixed integer programming (MIP) is characterizing when a disjunctive constraint of the form

$$z \in \bigcup_{i \in I} P_i \subset \mathbb{R}^n, \quad (1)$$

where  $P_i = \{z \in \mathbb{R}^n : A^i z \leq b^i\}$ , can be modeled as a binary integer program. Jeroslow and Lowe ([1][2][3]) showed that a necessary and sufficient condition is for  $\{P_i\}_{i \in I}$  to be a finite family of polyhedra with a common recession cone.

---

\* This research has been supported by NSF grant CMMI-0522485, AFOSR grant FA9550-07-1-0177 and Exxon Mobil Upstream Research Company.

\*\* The authors would like to thank Daniel Espinoza for pointing out the relation between SOS2 compatible functions and Gray codes.

Using results from disjunctive programming ([4,5,6,7,8,9]) they showed that, in this case, constraint (1) can be simply modeled as

$$A^i z^i \leq x_i b^i \quad \forall i \in I, \quad z = \sum_{i \in I} z^i, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0, 1\} \quad \forall i \in I. \quad (2)$$

The possibility of reducing the number of continuous variables in these models has been studied in [10,11,12], but the number of binary variables and extra constraints needed to model (1) has received little attention. However, it has been observed that a careful construction can yield a much smaller model than a naive approach. Perhaps the simplest example comes from the equivalence between general integer and binary integer programming (see for example page 12 of [13]). The requirement  $x \in [0, u] \cap \mathbb{Z}$  can be written in the form (1) by letting  $P_i := \{i\}$  for all  $i$  in  $I := [0, u] \cap \mathbb{Z}$  which, after some algebraic simplifications, yields a representation of the form (2) given by

$$z = \sum_{i \in I} i x_i, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0, 1\} \quad \forall i \in I. \quad (3)$$

This formulation has a number of binary variables that is linear in  $|I|$  and can be replaced by

$$z = \sum_{i=0}^{\lfloor \log_2 u \rfloor} 2^i x_i, \quad z \leq u, \quad x_i \in \{0, 1\} \quad \forall i \in \{0, \dots, \lfloor \log_2 u \rfloor\}. \quad (4)$$

In contrast to (3), (4) has a number of binary variables that is logarithmic in  $|I|$ . Although (4) appears in the mathematical programming literature as early as [14], and the possibility of modeling with a logarithmic number of binary variables and a linear number of constraints is studied in the theory of disjunctive programming (see for example [5]) and in [15], we are not aware of any other non-trivial formulations with a logarithmic number of binary variables and extra constraints.

The main objective of this work is to show that some well known classes of constraints of the form (1) can be modeled with a logarithmic number of binary variables and extra constraints. Although modeling with fewer binary variables and constraints might seem advantageous, a smaller formulation is not necessarily a better formulation (see for example section I.1.5 of [16]). More constraints might provide a tighter LP relaxation and more variables might do the same by exploiting the favorable properties of projection (see for example [17]). For this reason, we will also show that under some conditions our new formulations are as tight as any other mixed integer formulation, and we empirically show that they can provide a significant computational advantage.

The paper is organized as follows. In Section 2 we study the modeling of a class of hard combinatorial constraints. In particular we introduce the first formulations for SOS1 and SOS2 constraints that use only a logarithmic number of binary variables and extra constraints. In Section 3 we relate the modeling with

a logarithmic number of binary variables to branching and we introduce sufficient conditions for these models to exist. We then show that for a broad class of problems the new formulations are as tight as any other mixed integer programming formulation. In Section 4 we use the sufficient conditions to present a new formulation for non-separable piecewise linear functions of one and two variables that uses only a logarithmic number of binary variables and extra constraints. In Section 5 we show that the new models for piecewise linear functions of one and two variables can perform significantly better than the standard binary models. Section 6 gives some conclusions.

## 2 Modeling a Class of Hard Combinatorial Constraints

In this section we study a class of constraints of the form (1) in which the polyhedra  $P_i$  have the simple structure of only allowing some subsets of variables to be non-zero. Specifically, we study constraints over a vector of continuous variables  $\lambda$  indexed by a finite set  $J$  that are of the form

$$\lambda \in \bigcup_{i \in I} Q(S_i) \subset \Delta^J, \tag{5}$$

where  $I$  is a finite set,  $\Delta^J := \{\lambda \in \mathbb{R}_+^J : \sum_{j \in J} \lambda_j \leq 1\}$  is the  $|J|$ -dimensional simplex in  $\mathbb{R}^J$ ,  $S_i \subset J$  for each  $i \in I$  and

$$Q(S_i) = \{\lambda \in \Delta^J : \lambda_j \leq 0 \forall j \notin S_i\}. \tag{6}$$

Furthermore, without loss of generality we assume that  $\bigcup_{i \in I} S_i = J$ . Except for Theorem 3 our results easily extend to the case in which the simplex  $\Delta^J$  is replaced by a box in  $\mathbb{R}_+^J$ , but the restriction to  $\Delta^J$  greatly simplifies the presentation.

Disjunctive constraint (5) includes SOS1 and SOS2 constraints [18] over continuous variables in  $\Delta^J$ . SOS1 constraints on  $\lambda \in \mathbb{R}_+^n$  allow at most one of the  $\lambda$  variables to be non-zero which can be modeled by letting  $I = J = \{1, \dots, n\}$  and  $S_i = \{i\}$  for each  $i \in I$ . SOS2 constraints on  $(\lambda_j)_{j=0}^n \in \mathbb{R}_+^{n+1}$  allow at most two  $\lambda$  variables to be non-zero and have the extra requirement that if two variables are non-zero their indices must be adjacent. This can be modeled by letting  $I = \{1, \dots, n\}$ ,  $J = \{0, \dots, n\}$  and  $S_i = \{i - 1, i\}$  for each  $i \in I$ .

Mixed integer binary models for SOS1 and SOS2 constraints have been known for many years (see for example [19,20]), and some recent research has focused on branch-and-cut algorithms that do not use binary variables [21,22,23,24]. However, the incentive of being able to use state of the art MIP solvers (see for example the discussion in section 5 of [25]) makes binary models for these constraints very attractive (see for example [26,27,28,29]).

We first review a formulation for (5) with a linear number of binary variables and then we give a formulation with a logarithmic number of binary variables and a linear number of extra constraints. We then study how to obtain a formulation with a logarithmic number of variables and a logarithmic number of extra constraints and show that this can be achieved for SOS1 and SOS2 constraints.

The most direct way of formulating (5) as an integer programming problem is by assigning a binary variable for each set  $Q(S_i)$  and using formulation (2). After some algebraic simplifications this yields the formulation of (5) given by

$$\lambda \in \Delta^J, \quad \lambda_j \leq \sum_{i \in I(j)} x_i \quad \forall j \in J, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0, 1\} \quad \forall i \in I$$

where  $I(j) = \{i \in I : j \in S_i\}$ . This gives a formulation with  $|I|$  binary variables and  $|J| + 1$  extra constraints and yields the standard formulations for SOS1 and SOS2 constraints. (We consider  $\Delta^J$  as the original constraints and disregard the bounds on  $x$ ).

The following proposition shows that by using techniques from [15] we can obtain a formulation with  $\lceil \log_2 |I| \rceil$  binary variables and  $|I|$  extra constraints.

**Proposition 1.**  $B : I \rightarrow \{0, 1\}^{\lceil \log_2 |I| \rceil}$  .

$$\lambda \in \Delta^J, \quad \sum_{j \notin S_i} \lambda_j \leq \sum_{l \notin \sigma(B(i))} x_l + \sum_{l \in \sigma(B(i))} (1 - x_l) \quad \forall i \in I, \\ x_l \in \{0, 1\} \quad \forall l \in L(|I|) \quad (7)$$

$$\sigma(B) \qquad B \qquad L(r) := \{1, \dots, \lceil \log_2 r \rceil\}, \quad (5).$$

For SOS1 constraints, for which  $|I(j)| = 1$  for all  $j \in J$ , we can obtain the following alternative formulation of (5) which has  $\lceil \log_2 |I| \rceil$  binary variables and  $2\lceil \log_2 |I| \rceil$  extra constraints.

**Proposition 2.**  $B : I \rightarrow \{0, 1\}^{\lceil \log_2 |I| \rceil}$  .

$$\lambda \in \Delta^J, \quad \sum_{j \in J^+(l, B)} \lambda_j \leq x_l, \quad \sum_{j \in J^0(l, B)} \lambda_j \leq (1 - x_l), \quad x_l \in \{0, 1\} \quad \forall l \in L(|I|), \quad (8)$$

$$J^+(l, B) = \{j \in J : \forall i \in I(j) \quad l \in \sigma(B(i))\} \qquad J^0(l, B) = \{j \in J : \\ \forall i \in I(j) \quad l \notin \sigma(B(i))\}, \qquad 1$$

The following example illustrates formulation (8) for SOS1 constraints.

**Example 1.** Let  $J = \{1, \dots, 4\}$  and  $(\lambda_j)_{j=1}^4 \in \Delta^J$  be constrained to be SOS1 and let  $B^*(1) = (1, 1)^T$ ,  $B^*(2) = (1, 0)^T$ ,  $B^*(3) = (0, 1)^T$  and  $B^*(4) = (0, 0)^T$ . Formulation (8) for this case with  $B = B^*$  is

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0, 1\}, \quad \lambda_1 + \lambda_2 \leq x_1, \quad \lambda_3 + \lambda_4 \leq 1 - x_1, \quad \lambda_1 + \lambda_3 \leq x_2, \\ \lambda_2 + \lambda_4 \leq 1 - x_2.$$

Formulation (8) is valid for SOS1 constraints independent of the choice of  $B$ . In contrast, for SOS2 constraints, where  $|I(j)| = 2$  for some  $j \in J$ , formulation (8) can be invalid for some choices of  $B$ . This is illustrated by the following example.



**Example 2**

Let  $J = \{0, \dots, 4\}$  and  $(\lambda_j)_{j=0}^4 \in \Delta^J$  be constrained to be SOS2. Formulation (8) for this case with  $B = B^*$  is

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0, 1\}, \quad \lambda_0 + \lambda_1 \leq x_1, \quad \lambda_3 + \lambda_4 \leq 1 - x_1, \quad \lambda_0 \leq x_2, \\ \lambda_4 \leq 1 - x_2$$

which has  $\lambda_0 = 1/2, \lambda_2 = 1/2, \lambda_1 = \lambda_3 = \lambda_4 = 0, x_1 = x_2 = 1$  as a feasible solution that does not comply with SOS2 constraints. However, the formulation can be made valid by adding constraints

$$\lambda_2 \leq x_1 + x_2, \quad \lambda_2 \leq 2 - x_1 - x_2. \tag{9}$$

For any  $B$  we can always correct formulation (8) for SOS2 constraints by adding a number of extra linear inequalities, but with a careful selection of  $B$  the validity of the model can be preserved without the need for additional constraints.

**Definition 1 (SOS2 Compatible Function).**

$$B : \{1, \dots, n\} \rightarrow \{0, 1\}^{\lceil \log_2(n) \rceil} \quad \text{compatible} \quad \begin{matrix} 2 \\ B(i) \quad B(i+1) \end{matrix} \quad \text{ff}$$

**Theorem 1.**  $B$   $\tag{2}$   $\tag{8)}$   $\tag{2}$

The following example illustrates how an SOS2 compatible function yields a valid formulation.

**Example 2 (continued)**

Let  $B^0(1) = (1, 0)^T, B^0(2) = (1, 1)^T, B^0(3) = (0, 1)^T$  and  $B^0(4) = (0, 0)^T$ . Formulation (8) with  $B = B^0$  for the same SOS2 constraints is

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0, 1\} \\ \lambda_0 + \lambda_1 \leq x_1, \quad \lambda_3 + \lambda_4 \leq (1 - x_1) \tag{10}$$

$$\lambda_2 \leq x_2, \quad \lambda_0 + \lambda_4 \leq (1 - x_2). \tag{11}$$

An SOS2 compatible function can always be constructed and for each  $n \in \mathbb{Z}_+$  there are several SOS2 compatible functions. In fact, definition 1 is equivalent to requiring  $(B(i))_{i=1}^n$  to be a  $\text{ff}$  or (see for example [30]).

### 3 Branching and Modeling with a Logarithmic Number of Binary Variables and Constraints

The way in which formulation (8) is constructed does not provide a clear interpretation of the binary variables used, which makes it hard to extend to other combinatorial constraints. In this section we develop a more general scheme which is related to specialized branching schemes.

We can identify each vector in  $\{0, 1\}^{\lceil \log_2 |I| \rceil}$  with a leaf in a binary tree with  $\lceil \log_2 |I| \rceil$  levels in a way such that each component corresponds to a level and the value of that component indicates the selected branch in that level. Then, using function  $B$  we can identify each set  $Q(S_i)$  with a leaf in the binary tree and we can interpret each of the  $\lceil \log_2 |I| \rceil$  variables as the execution of a branching scheme on sets  $Q(S_i)$ . The formulations in Example 2 illustrates this idea.

In formulation (8) with  $B = B^0$  the branching scheme associated with  $x_1$  sets  $\lambda_0 = \lambda_1 = 0$  when  $x_1 = 0$  and  $\lambda_3 = \lambda_4 = 0$  when  $x_1 = 1$ , which is equivalent to the traditional SOS2 constraint branching of [18] whose dichotomy is fixing to zero variables to the “left of” (smaller than) a certain index in one branch and to the “right” (greater) in the other. In contrast, the scheme associated with  $x_2$  sets  $\lambda_2 = 0$  when  $x_2 = 0$  and  $\lambda_0 = \lambda_4 = 0$  when  $x_2 = 1$ , which is different from the traditional branching as its dichotomy can be interpreted as fixing variables in the “center” and on the “sides” respectively. If we use function  $B^*$  instead we recover the traditional branching. The drawback of the  $B^*$  scheme is that the second level branching cannot be implemented independently of the first one using linear inequalities. For  $B^0$  the branch alternatives associated with  $x_2$  are implemented by (11), which only include binary variable  $x_2$ . In contrast, for  $B^*$  one of the branching alternatives requires additional constraints (9) which involve both  $x_1$  and  $x_2$ .

This example illustrates that a sufficient condition for modeling (5) with a logarithmic number of binary variables and extra constraints is to have a binary branching scheme for  $\lambda \in \bigcup_{i \in I} Q(S_i)$  with a logarithmic number of dichotomies and for which each dichotomy can be implemented independently. This condition is formalized in the following definition.

**Definition 2.** (Independent Branching Scheme)  $\{L_k, R_k\}_{k=1}^d$   $L_k, R_k \subset J$   
 $d$  (5)

$$\bigcup_{i \in I} Q(S_i) = \bigcap_{k=1}^d (Q(L_k) \cup Q(R_k)).$$

This definition can then be used in the following theorem and immediately gives a sufficient condition for modeling with a logarithmic number of variables and constraints.

**Theorem 2.**  $\{Q(S_i)\}_{i \in I}$   $f_i$  (6)  
 $\{L_k, R_k\}_{k=1}^{\lceil \log_2(|I|) \rceil}$   $\lambda \in \bigcup_{i \in I} Q(S_i)$ .

$$\lambda \in \Delta^J, \quad \sum_{j \notin L_k} \lambda_j \leq x_k, \quad \sum_{j \notin R_k} \lambda_j \leq (1 - x_k),$$

$$x_k \in \{0, 1\} \quad \forall k \in \{1, \dots, \lceil \log_2(|I|) \rceil\} \quad (12)$$

(5)  $\lceil \log_2(|I|) \rceil$   $2 \lceil \log_2(|I|) \rceil$

Formulation (8) with  $B = B^0$  in Example 2 illustrates how an SOS2 compatible function induces an independent branching scheme for SOS2 constraints. In

general, given an SOS2 compatible function  $B : \{1, \dots, n\} \rightarrow \{0, 1\}^{\lceil \log_2(n) \rceil}$  the induced independent branching is given by  $L_k = J \setminus J^+(k, B)$ ,  $R_k = J \setminus J^0(k, B)$  for all  $k \in \{1, \dots, n\}$ .

Formulation (12) in Proposition 2 can be interpreted as a way of implementing a specialized branching scheme using binary variables. Similar techniques for implementing specialized branching schemes have been previously introduced for example, in [31] and [32], but the resulting models require at least a linear number of binary variables. To the best of our knowledge the first non-trivial independent branching schemes of logarithmic depth are the ones for SOS1 constraints from Proposition 2 and for SOS2 constraints induced by an SOS2 compatible function.

Formulation (12) can be obtained by algebraic simplifications from formulation (2) of (5) rewritten as the conjunction of two-term polyhedral disjunctions. Both the simplifications and the rewrite can result in a significant reduction in the tightness of the linear programming relaxation of (12) (see for example [5,10,11,12]). Fortunately, as the following propositions shows, the restriction to  $\Delta^J$  makes (12) as tight as any other mixed integer formulation for (5).

**Theorem 3.**  $P_\lambda \stackrel{(12)}{=} Q_\lambda \quad \lambda$

$$(5) \quad P_\lambda = \text{conv} \left( \bigcup_{i \in I} Q(S_i) \right) \quad P_\lambda \subseteq Q_\lambda.$$

Theorem 3 might no longer be true if we do not restrict to  $\Delta^J$ , but this restriction is not too severe as it includes a popular way of modeling piecewise linear functions. We explore this further in the following section.

### 4 Modeling Nonseparable Piecewise Linear Functions of Two Variables

In this section we use Theorem 2 to construct a model for non-separable piecewise linear functions of two variables that use a number of binary variables and extra constraints that is logarithmic in the number of linear pieces of the functions. Although some non-separable functions can be separated there are many practical reasons to avoid this separation (see for example the discussion on page 569 of [24]).

Imposing SOS2 constraints on  $(\lambda_j)_{j=0}^n \in \Delta^J$  with  $J = \{0, \dots, n\}$  is a popular way of modeling a one variable piecewise-linear function which is linear in  $n$  different intervals (see for example [22,23]). This approach has been extended to non-separable piecewise linear functions in [33,24,34,35]. For functions of two variables this approach can be described as follows.

We assume that for an even integer  $w$  we have a continuous function  $f : [0, w]^2 \rightarrow \mathbb{R}$  which we want to approximate by a piecewise linear function. A common approach is to partition  $[0, w]^2$  into a number of triangles and approximate  $f$  with a piecewise linear function that is linear in each triangle. One possible triangulation of  $[0, w]^2$  is the  $\mathbf{J}_1$  or ‘‘Union Jack’’ triangulation (see for example [36]) which is depicted in Figure 1(a) for  $w = 4$ . The  $\mathbf{J}_1$  triangulation

of  $[0, w]^2$  for any even integer  $w$  is simply obtained by adding copies of the 8 triangles shaded gray in Figure 1(a). This yields a triangulation with a total of  $2w^2$  triangles. We use this triangulation to approximate  $f$  with a piecewise

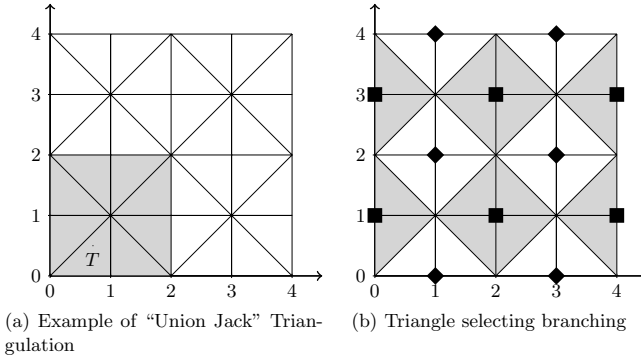


Fig. 1. Triangulations

linear function that we denote by  $g$ . Let  $I$  be the set of all the triangles of the  $\mathbf{J}_1$  triangulation of  $[0, w]^2$  and let  $S_i$  be the vertices of triangle  $i$ . For example, in Figure 1(a), the vertices of the triangle labeled  $T$  are  $S_T := \{(0, 0), (1, 0), (1, 1)\}$ . A valid model for  $g(y)$  (see for example [33,24,34]) is

$$\sum_{j \in J} \lambda_j = 1, \quad y = \sum_{j \in J} v_j \lambda_j, \quad g(y) = \sum_{j \in J} f(v_j) \lambda_j \tag{13a}$$

$$\lambda \in \bigcup_{i \in I} Q(S_i) \subset \Delta^J, \tag{13b}$$

where  $J := \{0, \dots, w\}^2$ ,  $v_j = j$  for  $j \in J$ . This model becomes a traditional model for one variable piecewise linear functions (see for example [22,23]) when we restrict it to one coordinate of  $[0, w]^2$ .

To obtain a mixed integer formulation of (13) with a logarithmic number of binary variables and extra constraints it suffices to construct an independent binary branching scheme of logarithmic depth for (13b) and use formulation (12). Binary branching schemes for (13b) with a similar triangulation have been developed in [34] and [24], but they are either not independent or have too many dichotomies. We adapt some of the ideas of these branching schemes to develop an independent branching scheme for the two-dimensional  $\mathbf{J}_1$  triangulation. Our independent branching scheme will basically select a triangle by forbidding the use of vertices in  $J$ . We divide this selection into two phases. We first select the square in the grid induced by the triangulation and we then select one of the two triangles inside this square.

To implement the first branching phase we use the observation made in [24,34] that selecting a square can be achieved by applying SOS2 branching to each

component. To make this type of branching independent it then suffices to use the independent SOS2 branching induced by an SOS2 compatible function. This results in the set of constraints

$$\sum_{s=0}^w \sum_{r \in J_2^+(l, B, w)} \lambda_{(r,s)} \leq x_{(1,l)}, \quad \sum_{s=0}^w \sum_{r \in J_2^0(l, B, w)} \lambda_{(r,s)} \leq 1 - x_{(1,l)},$$

$$x_{(1,l)} \in \{0, 1\} \quad \forall l \in L(w), \tag{14a}$$

$$\sum_{r=0}^w \sum_{s \in J_2^+(l, B, w)} \lambda_{(r,s)} \leq x_{(2,l)}, \quad \sum_{r=0}^w \sum_{s \in J_2^0(l, B, w)} \lambda_{(r,s)} \leq 1 - x_{(2,l)},$$

$$x_{(2,l)} \in \{0, 1\} \quad \forall l \in L(w), \tag{14b}$$

where  $B$  is an SOS2 compatible function and  $J_2^+(l, B, w)$ ,  $J_2^0(l, B, w)$  are the specializations of  $J^+(l, B)$ ,  $J^0(l, B)$  for SOS2 constraints on  $(\lambda_j)_{j=0}^w$ . Constraints (14a) implement the independent SOS2 branching for the first coordinate and (14b) do the same for the second one.

To implement the second phase we use the branching scheme depicted in Figure 1(b) for the case  $w = 4$ . The dichotomy of this scheme is to select the triangles colored white in one branch and the ones colored gray in the other. For general  $w$ , this translates to forbidding the vertices  $(r, s)$  with  $r$  even and  $s$  odd in one branch (square vertices in the figure) and forbidding the vertices  $(r, s)$  with  $r$  odd and  $s$  even in the other (diamond vertices in the figure). This branching scheme selects exactly one triangle of every square in each branch and induces the set of constraints

$$\sum_{(r,s) \in L} \lambda_{(r,s)} \leq x_0, \quad \sum_{(r,s) \in R} \lambda_{(r,s)} \leq 1 - x_0, \quad x_0 \in \{0, 1\}, \tag{15}$$

where  $L = \{(r, s) \in J : r \text{ is even and } s \text{ is odd}\}$  and  $R = \{(r, s) \in J : r \text{ is odd and } s \text{ is even}\}$ . This formulation is illustrated by the following example.

**Example 3.** Constraints (14)–(15) for  $w = 2$  are

$$\begin{aligned} \lambda_{(0,0)} + \lambda_{(0,1)} + \lambda_{(0,2)} &\leq x_{(1,1)}, & \lambda_{(2,0)} + \lambda_{(2,1)} + \lambda_{(2,2)} &\leq 1 - x_{(1,1)} \\ \lambda_{(0,0)} + \lambda_{(1,0)} + \lambda_{(2,0)} &\leq x_{(2,1)}, & \lambda_{(0,2)} + \lambda_{(1,2)} + \lambda_{(2,2)} &\leq 1 - x_{(2,1)} \\ \lambda_{(0,1)} + \lambda_{(2,1)} &\leq x_0, & \lambda_{(1,0)} + \lambda_{(1,2)} &\leq 1 - x_0. \end{aligned}$$

A portion of the associated branching scheme is shown in Figure 2. The shaded triangles inside the nodes indicates the triangles forbidden by the corresponding assignment of the binary variables. The restriction to the first coordinate of  $[0, w]^2$  yields a logarithmic formulation for piecewise linear functions of one variable that only uses one of the SOS2 branchings and does not use the triangle selecting branching. Furthermore, under some mild assumptions, the model can be extended to non-uniform grids by selecting different values of  $v_j$  and to functions of 3 variables as well.

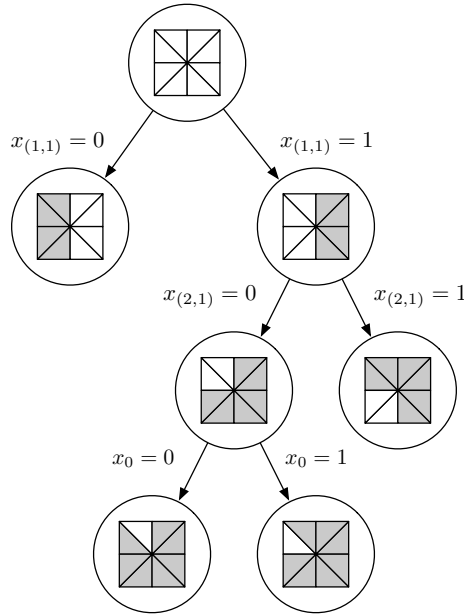


Fig. 2. Partial B&B tree from Example 3

## 5 Computational Results

In this section we computationally test the logarithmic models for piecewise linear functions of one and two variables against some other existing models. For a set of transportation problems with piecewise linear cost functions, the logarithmic models provide a significant advantage in almost all of our experiments.

We denote the model for piecewise linear functions of one and two variables from section 4 by (Log). From the traditional models we selected the one usually denoted as the incremental model. This model for one variable functions appears as early as [19,37,20] and it has been recently shown to have favorable integrality and tightness properties [26,28,29]. The incremental model was extended to functions of several variables in [35]. We denote this model by (Inc). We also include two models that are based on independent branching schemes of linear depth. The first model is based on the independent branching scheme for SOS2 constraints on  $(\lambda_j)_{j=0}^n$  given by  $L_k = \{k, \dots, n\}$ ,  $R_k = \{0, \dots, k\}$  for every  $k \in \{1, \dots, n - 1\}$ . This formulation has been independently developed in [32] and is currently defined only for functions of one variable. We denote this model by (LB1). The second model is based on an independent branching defined in [24, p. 573]. This branching scheme is defined for any triangulation and it has one branch for every vertex in the triangulation. In particular for piecewise linear functions of one variable with  $k$  intervals or segments the scheme has  $k + 1$  branches and for piecewise linear functions on a  $k \times k$  grid it has  $(k + 1)^2$

branches. We denote the model by (LB2). We also tested some other piecewise linear models, but do not report results for them since they did not significantly improve the worst results reported here. All models were generated using Ilog Concert Technology and solved using CPLEX 9 on a dual 2.4GHz Linux workstation with 2GB of RAM. Furthermore, all tests were run with a time limit of 10000 seconds.

The first set of experiments correspond to piecewise linear functions of one variable for which we used the transportation models from [25]. We selected the instances with 10 supply and 10 demand nodes and for each of the 5 available instances we generated several randomly generated objective functions. We generated a separable piecewise linear objective function given by the sum of concave non-decreasing piecewise linear functions of the flow in each arc. For each instance and number of segments we generated 20 objective functions to obtain a total of 100 instances for each number of segments. Tables 1(a), 1(b) and 1(c) show the minimum, average, maximum and standard deviation of the solve times in seconds for 4, 8 and 16 segments. The tables also shows the number of times the solves failed because the time limit was reached and the number of times each formulation had the fastest solve time. As a final test for the one variable functions we tested the 3 best models on 100 instances with functions with 32 segments. Table 1(d) presents the statistics for these instances. For 16 and 32 segments we excluded the “wins” row as (Log) had the fastest solve times for every instance. The next set of experiments correspond to piecewise linear

**Table 1.** Solve times for one variable functions [s]

stat	(Log)	(LB1)	(LB2)	(Inc)	stat	(Log)	(LB1)	(LB2)	(Inc)
min	0	0	1	1	min	1	0	1	0
avg	2	2	4	3	avg	8	19	88	44
max	9	9	27	16	max	44	162	1171	245
std	1	1	3	2	std	8	19	147	36
fails	0	0	0	0	fails	0	0	0	0
wins	72	27	0	1	wins	98	1	1	0
(a) 4 segments.					(b) 8 segments.				
stat	(Log)	(LB1)	(LB2)	(Inc)	stat	(Log)	(LB1)	(Inc)	
min	1	13	15	46	min	3	113	182	
avg	19	127	3561	374	avg	33	880	1445	
max	83	652	10000	1859	max	174	10000	8580	
std	17	105	3912	338	std	33	1289	1327	
fails	0	0	21	0	fails	0	1	0	
(c) 16 segments.					(d) 32 segments.				

functions of two variables and we again used the  $10 \times 10$  transportation models from [25]. In this case we took two copies of the same transportation model for each instance. We used an objective function which is the sum over all the arcs in the original transportation problem of non-separable two variable piecewise

linear functions of the flows in the two copies of the arc. For each arc we generated the corresponding two variable piecewise linear function by triangulating a domain of the form  $[0, w]^2$  as described in section 4 with a  $8 \times 8$  segment grid to obtain a total of 128 triangles with 81 vertices. We then interpolated on this grid the functions of the two flows  $x_e, x_{e'}$  given by  $\sqrt{(a_1x_e + b_1)(a_2x_{e'} + b_2)}$  for  $a_1, b_1, a_2, b_2 \in \mathbb{R}_+$  randomly generated independently for each arc. In addition, we eliminated the supply constraints from the two copies of the transportation problem to make the instances easier to solve. These problems were not created with a realistic application in mind and are just a simple extension of the problems in the previous set designed to include two variable non-separable functions. We again generated 20 objective functions for each of the original instances for a total of 100 instances. We excluded formulation (LB1) in this second set of tests as it is only valid for functions of one variable. Table 2(a) shows the statistics for this set of instances. As a final experiment we generated a new set of problems using a  $16 \times 16$  grid for the interpolation obtaining a total of 512 triangles and 289 vertices. For these instances we only used formulations (Log) and (LB2). Table 2(b) shows the statistics for this last set of instances. It is clear that one

**Table 2.** Solve times for two variable functions on a  $8 \times 8$  and  $16 \times 16$  grids [s]

stat	(Log)	(LB2)	(Inc)		stat	(Log)	(LB2)
min	1	3	95		min	5	22
avg	11	78	3521		avg	374	2910
max	102	967	10000		max	10000	10000
std	15	140	3648		std	1057	3444
fails	0	0	19		fails	1	11
wins	99	1	0		wins	98	2
(a) $8 \times 8$ grid.					(b) $16 \times 16$ grid.		

of the advantages of the (Log) formulation is that it is smaller than the other formulations while retaining favorable tightness properties. In addition, formulation (Log) effectively transforms CPLEX’s binary variable branching into a specialized branching scheme for piecewise linear functions. This allows formulation (Log) to combine the favorable properties of specialized branching schemes and the technology in CPLEX’s variable branching. This last property is what probably allows (LB1) and (LB2) to outperform (Inc) too. In this regard we would like to emphasise the fact that all models tested are pure mixed integer programming problems (i.e. they do not include high level SOS2 constraints). Although CPLEX allows SOS2 high level descriptions and can use specialized SOS2 branching schemes that do not use binary variables the performance of these features for CPLEX 9 was inferior to most binary models we tested (including all for which results are presented here). Preliminary tests with CPLEX 11 show that these features have been considerably improved, which could make them competitive with the binary models. It is clear that formulation (Log) is superior to all of the others and that its advantage increases as the number of segments grows.



## 6 Conclusions

We have introduced a technique for modeling hard combinatorial problems with a mixed 0-1 integer programming formulation that uses a logarithmic number of binary variables and extra constraints. It is based on the concept of independent branching which is closely related to specialized branching schemes for combinatorial optimization. Using this technique we have introduced the first binary formulations for SOS1 and SOS2 constraints and for one and two variable piecewise linear functions that use a logarithmic number of binary variables and extra constraints. Finally, we have illustrated the usefulness of these new formulations by showing that for one and two variable piecewise linear functions they provide a significant computational advantage.

There are still a number of unanswered questions concerning necessary and sufficient conditions for the existence of formulations with a logarithmic number of binary variables and extra constraints. Simple examples show that it may not always be possible to obtain such a model. Moreover, if we allow the formulation to have a number of binary variables and extra constraints whose asymptotic growth is logarithmic our sufficient conditions do not seem to be necessary. Consider cardinality constraints that restrict at most  $K$  components of  $\lambda \in [0, 1]^n$  to be non-zero. This constraint does not satisfy the sufficient conditions but it does have a formulation with a number of variables and constraints of logarithmic order. We can write cardinality constraints in the form (5) by letting  $J = \{1, \dots, n\}$ ,  $I = \{1, \dots, m\}$  for  $m = \binom{n}{K}$  and  $\{S_j\}_{j=1}^m$  be the family of all subsets of  $J$  such that  $|S_i| = K$ . The traditional formulation for cardinality constraints is [19,20]

$$\sum_{j=1}^n x_j \leq K; \quad \lambda_j \in [0, 1], \quad \lambda_j \leq x_j, \quad x_j \in \{0, 1\} \quad \forall j \in J. \quad (16)$$

Let  $n$  be an even number. By choosing  $K = n/2$ , which is the non-trivial cardinality constraint with the largest number of sets  $S_i$ , we can use the fact that for  $K = n/2$  we have  $n \leq 2 \log_2 \left( \binom{n}{K} \right)$  to conclude that (16) has  $O(\log_2(|I|))$  binary variables and extra constraints.

## References

1. Jeroslow, R.G.: Representability in mixed integer programming 1: characterization results. *Discrete Applied Mathematics* 17, 223–243 (1987)
2. Jeroslow, R.G., Lowe, J.K.: Modeling with integer variables. *Mathematical Programming Study* 22, 167–184 (1984)
3. Lowe, J.K.: Modelling with Integer Variables. PhD thesis, Georgia Institute of Technology (1984)
4. Balas, E.: Disjunctive programming. *Annals of Discrete Mathematics* 5, 3–51 (1979)
5. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic and Discrete Methods* 6, 466–486 (1985)

6. Balas, E.: Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics* 89, 3–44 (1998)
7. Blair, C.: 2 rules for deducing valid inequalities for 0-1 problems. *SIAM Journal on Applied Mathematics* 31, 614–617 (1976)
8. Jeroslow, R.G.: Cutting plane theory: disjunctive methods. *Annals of Discrete Mathematics* 1, 293–330 (1977)
9. Sherali, H.D., Shetty, C.M.: Optimization with Disjunctive Constraints. Lecture Notes in Economics and Mathematical Systems, vol. 181. Springer, Heidelberg (1980)
10. Balas, E.: On the convex-hull of the union of certain polyhedra. *Operations Research Letters* 7, 279–283 (1988)
11. Blair, C.: Representation for multiple right-hand sides. *Mathematical Programming* 49, 1–5 (1990)
12. Jeroslow, R.G.: A simplification for some disjunctive formulations. *European Journal of Operational Research* 36, 116–121 (1988)
13. Garfinkel, R.S., Nemhauser, G.L.: *Integer Programming*. John Wiley & Sons, Inc., Chichester (1972)
14. Watters, L.J.: Reduction of integer polynomial programming problems to zero-one linear programming problems. *Operations Research* 15, 1171–1174 (1967)
15. Ibaraki, T.: Integer programming formulation of combinatorial optimization problems. *Discrete Mathematics* 16, 39–52 (1976)
16. Nemhauser, G.L., Wolsey, L.A.: *Integer and combinatorial optimization*. Wiley-Interscience, Chichester (1988)
17. Balas, E.: Projection, lifting and extended formulation in integer and combinatorial optimization. *Annals of Operations Research* 140, 125–161 (2005)
18. Beale, E.M.L., Tomlin, J.A.: Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In: Lawrence, J. (ed.) *OR 69. Proceedings of the fifth international conference on operational research*, pp. 447–454. Tavistock Publications (1970)
19. Dantzig, G.B.: On the significance of solving linear-programming problems with some integer variables. *Econometrica* 28, 30–44 (1960)
20. Markowitz, H.M., Manne, A.S.: On the solution of discrete programming-problems. *Econometrica* 25, 84–110 (1957)
21. de Farias Jr., I.R., Johnson, E.L., Nemhauser, G.L.: Branch-and-cut for combinatorial optimization problems without auxiliary binary variables. *The Knowledge Engineering Review* 16, 25–39 (2001)
22. Keha, A.B., de Farias, I.R., Nemhauser, G.L.: Models for representing piecewise linear cost functions. *Operations Research Letters* 32, 44–48 (2004)
23. Keha, A.B., de Farias, I.R., Nemhauser, G.L.: A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. *Operations Research* 54, 847–858 (2006)
24. Martin, A., Moller, M., Moritz, S.: Mixed integer models for the stationary case of gas network optimization. *Mathematical Programming* 105, 563–582 (2006)
25. Vielma, J.P., Keha, A.B., Nemhauser, G.L.: Nonconvex, lower semicontinuous piecewise linear optimization. *Discrete Optimization* 5, 467–488 (2008)
26. Croxton, K.L., Gendron, B., Magnanti, T.L.: A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science* 49, 1268–1273 (2003)

27. Magnanti, T.L., Stratila, D.: Separable concave optimization approximately equals piecewise linear optimization. In: Bienstock, D., Nemhauser, G.L. (eds.) IPCO 2004. LNCS, vol. 3064, pp. 234–243. Springer, Heidelberg (2004)
28. Padberg, M.: Approximating separable nonlinear functions via mixed zero-one programs. *Operations Research Letters* 27, 1–5 (2000)
29. Sherali, H.D.: On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions. *Operations Research Letters* 28, 155–160 (2001)
30. Wilf, H.S.: Combinatorial algorithms—an update. In: CBMS-NSF regional conference series in applied mathematics. Society for Industrial and Applied Mathematics, vol. 55 (1989)
31. Appleget, J.A., Wood, R.K.: Explicit-constraint branching for solving mixed-integer programs. In: Laguna, M., González Velarde, J.L. (eds.) Computing tools for modeling, optimization, and simulation: interfaces in computer science and operations research. *Operations research / computer science interfaces series*, vol. 12, pp. 245–261. Kluwer Academic, Dordrecht (2000)
32. Shields, R.: Personal communication (2007)
33. Lee, J., Wilson, D.: Polyhedral methods for piecewise-linear functions I: the lambda method. *Discrete Applied Mathematics* 108, 269–285 (2001)
34. Tomlin, J.: A suggested extension of special ordered sets to non-separable non-convex programming problems. In: Hansen, P. (ed.) *Studies on Graphs and Discrete Programming*. *Annals of Discrete Mathematics*, vol. 11, pp. 359–370. North Holland, Amsterdam (1981)
35. Wilson, D.: Polyhedral methods for piecewise-linear functions. PhD thesis, University of Kentucky (1998)
36. Todd, M.J.: Union jack triangulations. In: Karamardian, S. (ed.) *Fixed Points: algorithms and applications*, pp. 315–336. Academic Press, London (1977)
37. Dantzig, G.B.: *Linear Programming and Extensions*. Princeton University Press, Princeton (1963)

# Computing with Multi-row Gomory Cuts

Daniel G. Espinoza\*

Universidad de Chile, Department of Industrial Engineering,  
Santiago RM, 837-0439, Chile  
daespino@dii.uchile.cl

**Abstract.** Cutting planes for mixed integer problems (MIP) are nowadays an integral part of all general purpose software to solve MIP. The most prominent, and computationally significant, class of general cutting planes are Gomory mixed integer cuts (GMI). However finding other classes of general cuts for MIP that work well in practice has been elusive. Recent advances on the understanding of valid inequalities derived from the infinite relaxation introduced by Gomory and Johnson for mixed integer problems, has opened a new possibility of finding such an extension. In this paper, we investigate the computational impact of using a subclass of minimal valid inequalities from the infinite relaxation, using different number of tableau rows simultaneously, based on a simple separation procedure. We test these ideas on a set of MIPs, including MIPLIB 3.0 and MIPLIB 2003, and show that they can improve MIP performance even when compared against commercial software performance.

## 1 Introduction

The most successful approach to solve general MIP today is branch and cut, where general cutting planes are a crucial factor for the overall performance. After the great success in the 90's of using general purposes cutting planes such as GMI cuts [9,5], a great deal of research was devoted to extend those ideas to find other families of general cuts that consistently outperform GMI cuts. However, results have been mixed, and although there are several extensions that in theory are at least as good as GMI cuts, in practice they do not seem to offer much advantage. Most of the extensions have focused on deriving inequalities from the master cyclic group problem introduced by Gomory and Johnson [11], which look at a single constrained problem.

The theoretical importance of looking at multi-row relaxations has been proved in a number of works. For instance, Cook et al. [6], show an example with infinite Chvátal-Gomory rank (i.e. obtaining the convex hull of the integer points by adding inequalities derived from one row relaxations is impossible). Andersen et al. [3], prove that by looking at inequalities generated from two row relaxations, the convex hull of the Cook-Kannan-Schrijver example, can be

---

\* This research was partially funded by FONDECYT grant 1070749 and by ICM grant P05-004F.

obtained by adding a single cut. This situation is extended to higher dimensions in the work of Yanjun Li and Jean-Philippe P. Richard.

An interesting recent development has been the work of Andersen et al. [3], Cornuéjols and Borozan [7] and Gomory [10]; who have proposed to look at the so-called infinite relaxation problem, which was also introduced by Gomory and Johnson [11], and where several constraint are considered at the same time. The novelty of this relaxation is that it works on a continuous relaxation, and looks at an arbitrary number of tableau rows at the same time. Cornuéjols and Borozan [7] show that any minimal valid inequality for the relaxation can be related to maximal, convex, lattice-free polyhedrons; thus identifying inequalities with simple geometrical entities.

To the best of our knowledge, no computational test of the impact of using cuts derived from this relaxation have been published. The main contribution of this paper is to show that they are also very valuable in practice, not only improving the root LP integrality GAP ( $GAP_{LP}$ ) closed at the root node, but also in speeding-up the overall branch and cut performance when compared with CPLEX [12] defaults.

The rest of the paper is organized as follows. Section 2 presents the definition and basic results related to the infinite relaxation. Section 3 presents the basic computational problems, tradeoffs, and main ideas that guided the implementation, and also some further ideas to speed-up cut-generation and possible alternative choices. Section 4 explain our experiments, settings, and results.

## 2 The Infinite Relaxation

Consider a general mixed integer program (MIP)

$$\begin{aligned}
 &\min cx \\
 &s.t. Ax = b \\
 &\quad x_i \in \mathbb{Z} \forall i \in I \\
 &\quad x_i \geq 0 \forall i = 1, \dots, n,
 \end{aligned} \tag{1}$$

where  $I \subseteq \{1, \dots, n\}$ ,  $A \in \mathbb{Q}^{m \times n}$  is of full row rank,  $c \in \mathbb{Q}^n$ ,  $b \in \mathbb{Q}^m$ , and  $x \in \mathbb{Q}^n$ . Branch and cut algorithms start by solving

$$\begin{aligned}
 &\min cx \\
 &s.t. Ax = b \\
 &\quad x_i \geq 0 \forall i = 1, \dots, n,
 \end{aligned} \tag{2}$$

the LP relaxation of (1), and obtain an optimal basic solution of the form

$$x_B = f + \sum_{j \in N} r^j x_j, \tag{3}$$

where  $B$  is the set of basic variables satisfying  $B \subseteq \{1, \dots, n\}$ ,  $|B| = m$ ,  $N$  is the set of non-basic variables defined as  $N = \{1, \dots, n\} \setminus B$ , and  $f, r^j \in \mathbb{Q}^m, \forall j \in N$ ,

$f \geq 0$ . The basic solution is  $x^* = (x_B, x_N) = (f, 0)$ , and is an optimal solution to (1) if and only if  $x_i^* \in \mathbb{Z}, \forall i \in I' = I \cap B$ . If not, then one might try to find a valid inequality cutting off  $x^*$  from the feasible region of (2).

One possibility is to consider the following relaxation of (1):

$$\begin{aligned} z &= f + \sum_{i \in N \cap I} (r^i - a^i) s_i + \sum_{i \in N \setminus I} r^i s_i, \\ z &\in \mathbb{Q}^{I'}, \\ s &\geq 0, \end{aligned} \tag{4}$$

where we drop all basic continuous variables, drop the non-negativity constraints on the basic integer variables, and where  $a^i \in \mathbb{Z}^{I'}, \forall i \in I \cap N, z = x_{I'} - \sum_{i \in I \cap N} a^i s_i$ , and then relax  $s_i$  to be continuous. This relaxation was considered in [3,10] for the case  $|I'| = 2$ .

Gomory and Johnson [11] suggested relaxing (4) to an infinite-dimensional space; following the notation in [7]; it can be described as:

$$\begin{aligned} x &= f + \sum_{finite} r s_r \\ x &\in \mathbb{Z}^q \\ s &\geq 0 \end{aligned} \tag{5}$$

where  $s_r$  is defined for every  $r \in \mathbb{Q}^q$ , and  $\sum_{finite}$  means that  $|r : s_r > 0| \in \mathbb{N}$ , i.e.  $s$  has finite support. This is called the  $f_i$  and is denoted by  $R_f$ , where the feasible solutions of  $R_f$  are vectors  $(x, s)$  with finite support satisfying (5). Note that any valid inequality for (5) yields a valid inequality for (1).

Borožan and Cornuéjols [7] studied minimal valid inequalities for (5), proving the following theorem:

**Theorem 1 (Minimal Valid Inequalities for  $R_f$  [7]).**  $f \notin \mathbb{Z}^q$ ,  $ff(f, 0)$

$$\begin{aligned} &\sum_{finite} \psi(r) s_r \geq 1. \\ &B_\psi = \{x \in \mathbb{Q}^p : \psi(x - f) \leq 1\}, \quad B_\psi \\ &f \in B_\psi. \\ &\psi \quad f_i \quad , \quad \psi \\ &2^q \\ &\psi \quad f_i \quad , \quad f \quad B_\psi \quad B_\psi \\ &2^q \end{aligned}$$

One of the consequences of Theorem 1 is that it allow us to identify minimal valid inequalities  $\psi$  with the set  $B_\psi$ , providing a simple geometrical interpretation for them. We use this interpretation to chose a sub-family of minimally valid inequalities for (5). It is worth mentioning that the results of Theorem 1 where simultaneously conjectured (and partially proved) by Gomory in [10].

### 3 Selecting a Subclass of Valid Inequalities, and Separating Them

Thanks to the results in [7], the problem of finding minimal valid inequalities for (5), can be reduced to the problem of looking at maximal lattice-free polyhedra in  $\mathbb{Q}^q$ , where the lattice is just  $\mathbb{Z}^q$ . Although the characterization of all maximal lattice-free convex sets in the plane is known [15], such a characterization is unknown for arbitrary dimensions.

For general dimension  $q$ , we can define the following full-dimensional maximal lattice-free bounded convex sets:

1. The simplex defined by the points  $\{0, \pm ke_i : i = 1, \dots, q\}$ .
2. The set  $B_a = \frac{1}{2} + \{x : a^\delta x \leq a^\delta \delta, \forall \delta \in \Delta\}$  where  $\Delta = \{\{-\frac{1}{2}, \frac{1}{2}\}^q\}$ ,  $0 < a^\delta \in \mathbb{Q}^q$  and  $a_i^\delta \neq 0, \forall i = 1, \dots, q, \delta \in \Delta$ .

These two classes of sets represent the two extremes in terms of number of facets; in the first family, each set has  $q + 1$  facets, while in the second family, each set has  $2^q$  facets. Note also that each of their facets contains an integer point in their relative interior, thus they define minimal valid inequalities for (5).

For the case  $q = 2$ , Cornuéjols and Margot [8] proved that all simplex-related sets (called triangle inequalities in [3]) are facet defining for  $R_f$ , but that not all  $B_a$  sets define facets of  $R_f$ . However, is easy to see that there exist an arbitrarily small perturbation  $\varepsilon$  of  $a$ , such that  $B_{a+\varepsilon}$  defines a facet of  $R_f$ . This observation, and the limited numerical precision of floating point representation, justify, from a practical point of view, overlooking the fact that some  $B_a$  do not define a facet of  $R_f$  for  $q = 2$ . Although a similar result for arbitrary  $q$  is unknown, it seems reasonable to conjecture that related arguments should show the importance of the sets  $B_a$  in general.

This gives us a wide range of possible sets  $B$  to choose from. However, if we restrict ourselves to sets that are symmetric with respect to each coordinate axis, then, the only possible choice for  $B$  is the family  $B_a$ , where all  $a^\delta \equiv a$  for some  $a \in \mathbb{Q}_+^q$  (we assume that  $0 \notin \mathbb{Q}_+$ ). This restriction implies that the resulting cut should be invariant under multiplication of -1 to any constraint in (5).

From this point on, we focus on this kind of lattice-free sets. We assume that  $f \in (0, 1)$ , and define  $f' = f - \frac{1}{2}e$ , where  $e$  is the vector of all ones. With this,  $\psi_a$  (the function related to  $B_a$ ) can be defined as follows:

$$\psi_a(r) = \begin{cases} 0 & \text{if } r = 0 \\ 2 \max_{\delta \in \Delta} \left\{ \frac{\phi_\delta(a,r)}{a_o - \phi_\delta(a,f')} : \phi_\delta(a,r) > 0 \right\} & \text{if } r \neq 0 \end{cases}, \tag{6}$$

where  $\phi_\delta(a, b) = \sum (a_i \delta_i b_i : i = 1, \dots, q)$  and  $a_o = \frac{1}{2}a \cdot e$ .

Note that the amount of work to compute  $\psi_a(r)$  is exponential in  $q$ , however, one can speed up the process by using gray-code enumeration of  $\Delta$ . In our code we use Knuth’s loopless gray binary generation (LGBG) algorithm [13] to speed-up the computation of  $\psi_a(r)$ , moreover, we compute  $\psi_a(r)$  for all required  $r$  at the same time. Additional speed gains can be achieved by noting that in LGBG, index  $i$  changes its value exactly  $2^{q-i}$  times during the algorithm, thus sorting each row in

decreasing order by number of non-zeros should decrease the amount of total work. Finally, another factor of two can be gained by maintaining a list of  $r : \phi_\delta(a, r) > 0$ .

Another problem is to choose appropriate vectors  $a$ . One possibility is to use branching pseudo-cost values (see [11, 14] for details on pseudo-cost branching) to define the  $a_i$ . Instead, we use  $a_i = 1, \forall i = 1, \dots, q$ , but select the fractional variables to consider using branching pseudo-cost information.

For integer non-basic variables we select  $a^i$  in (4), such that  $r^{i'} = r^i - a^i \in [-\frac{1}{2}, \frac{1}{2}]^q$ , in the hope of obtaining small coefficients for  $\psi_a(r^{i'})$ . Note, that such a choice may not be the best possible.

To improve numerical stability of the cuts, we choose from fractional variables that are away from the nearest integer by at least  $2^{-12}$ ; also, the ratio between the smallest and largest absolute value in the cut should not exceed  $2^{15}$ ; if the minimum non-zero absolute coefficient in the cut ( $|c|_{min}$ ) is above one, we divide the resulting cut by  $|c|_{min}$ ; we discard cuts whose violation is below  $2^{-10}$ ; finally, we add cuts only at the root node of the branch and cut run. The code is available at <http://dii.uchile.cl/~daespino>.

## 4 Computational Results

Our computing environment is a Linux 2.6.22 machine with 1Gb. of RAM, with a 3GHz. Intel Pentium 4 CPU with 1Mb of cache; all the code is written in C, and was compiled with GCC 4.2.0 with optimization flags -O3.

Our cutting scheme was embedded as a cut-callback in CPLEX 10.2, and is called after CPLEX has added its own cuts. In every call we add at most one cut, but the procedure may be called many times during the optimization process. Our procedure adds cuts only at the root node. We compare our results against CPLEX defaults, with pre-processing turned on; this include automatic generation of clique cuts, lifted cover cuts, implied bound cuts, lifted flow cover cuts, flow path cuts and Gomory fractional cuts.

Our test set of MIP instances contains all MIPLIB 3.0 [4], MIPLIB 2003 [2], and other problems from the literature. The full test set contained 173 problems, from where we discard all problems (29) where the  $GAP_{LP}$  after solving with CPLEX 10.2 defaults was below 0.1%; then we discard all problems (34) CPLEX could solve to optimality within five seconds of CPU time; then we discarded all problems (15) where neither CPLEX nor our cutting procedure could improve the root LP bound [1]; finally, we discarded all problems (8) where our cutting procedure could not add any cut [2]. This reduced our test-set to 87 problems.

We tested six configurations, CPLEX defaults (C0), and the configurations T2N5, T5N5, T10N5, T10N1k and T15N1k [3], where T N signifies the adding of up to cuts generated using up to tableau rows. The first four configurations where run with a time limit of one hour, while the last two configurations where run with a time limit of 20 minutes.

<sup>1</sup> Table 1 contain the list of all such problems.

<sup>2</sup> Table 2 contain the list of all such problems.

<sup>3</sup> 1k stand for 1000, i.e. a thousand.

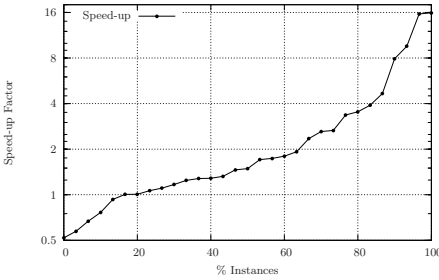


**Table 1.** Problems where neither CPLEX nor our procedure could improve the root LP value

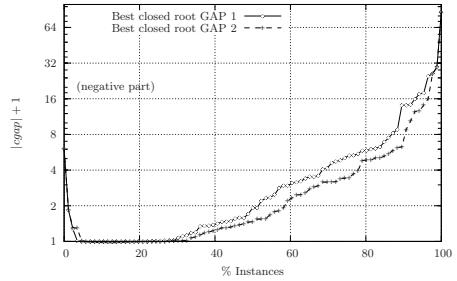
berlin_5_8_0	CMS750_4	glass4	marketshare1	marketshare2
neos19	neos818918	neos823206	net12	noswot
p2m2p1m1p0n100	railway_8_1_0	rd-rplusc-21	usAbbrv.8.25_70	van

**Table 2.** Problems where our procedure could not add any cut to the problem

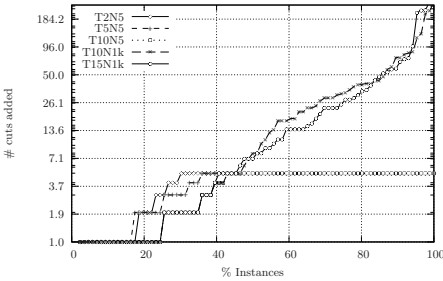
bg512142	dano3mip	dano3mip.pre	dg012142
harp2	mod011	momentum3	neos4



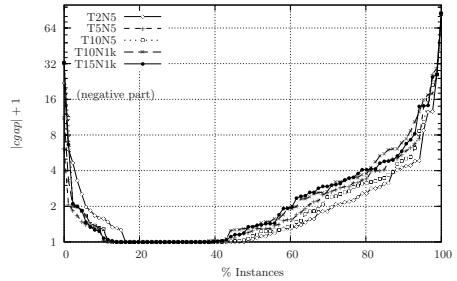
**Fig. 1.** Overall speed-up



**Fig. 2.** Best closed  $GAP_{LP}$



**Fig. 3.** Cuts added by configuration



**Fig. 4.** Closed  $GAP_{LP}$  by configuration

Tables 3, 4, 5 present our computational results over the reduced test-set. The first column indicates the problem name; the following six columns, give the root LP bound and the running time for the corresponding configuration; finally, the last column, has the optimal/best known solution for instance, and then the maximum of the closed  $GAP_{LP}$ , defined as  $(Z_{LP} - Z_{CPX}) / (Z_{IP} - Z_{CPX})$  <sup>4</sup> over

<sup>4</sup> where  $Z_{LP}$  is the root LP value for the configuration,  $Z_{IP}$  is the value of the optimal/best solution known for the problem, and  $Z_{CPX}$  is the value of the root LP obtained with CPLEX defaults.

**Table 3.** Results over reduced test set, part I

Configuration	C0		T2N5		T5N5		T10N5		T10N1k		T15N1k		Results
	Root LP	Time (s)	Root LP	Time (s)	Root LP	Time (s)	Root LP	Time (s)	Root LP	Time (s)	Root LP	Time (s)	
a1c1s1	9770.47		9797.84		9854.74		9799.46		9799.46		9823.78		11503.4
	3592.91		3593.45		3592.71		3594.60		1193.85		1194.55		4.86%/4.86%
A1C1S1	9804.58		9862.15		9869.86		9838.56		9838.56		9829.37		11503.4
	3587.77		3600.33		3594.44		3584.48		1195.51		1196.85		3.84%/3.84%
A2C1S1	9468.86		9394.16		9394.16		9320.6		9320.6		9385.61		10938.8
	3590.22		3598.84		3598.90		3591.90		1195.77		1194.16		-5.08%/-5.08%
aflow30a	1075.14		1081.59		1074.58		1074.58		1074.58		1075.27		1158
	66.62		55.56		45.79		50.67		48.33		50.86		7.79%/7.79%
aflow40b	1059.61		1058.88		1059.61		1062.55		1062.55		1062.94		1168
	3585.42		3563.11		3597.93		3596.80		1193.03		1194.28		3.07%/2.71%
air04	55645.7		55664.6		55660.4		55663.3		55667		55664.9		56137
	42.33		75.30		48.47		64.48		154.15		405.37		4.33%/3.86%
air05	25957.8		25972		25970.5		25973.7		25978.6		25973.7		26374
	31.48		83.59		79.81		66.11		500.72		295.92		5.00%/3.83%
B1C1S1	15496.3		15471.8		15463.4		15529		15529		15451.8		24881.7
	3577.55		3586.90		3586.32		3579.90		1191.87		1192.43		0.35%/0.35%
B2C1S1	16012.5		16112.2		16165.8		15969.5		15969.5		16213		26282.5
	3587.55		3590.03		3588.81		3589.90		1190.73		1192.06		1.95%/1.49%
bc1	2.47		2.47		2.47		2.47		2.48		2.47		3.34
	274.66		493.91		436.80		880.40		923.93		585.60		0.47%/0.26%
bell3a	873792		873792		873792		873792		873792		873792		878430
	7.09		7.17		7.53		7.22		7.54		7.28		0.00%/0.00%
biella1	3.0605e+06		3.0605e+06		3.0605e+06		3.0605e+06		3.0605e+06		3.0605e+06		3.0678e+06
	3590.18		3591.99		3532.93		3593.41		1191.94		1195.15		-0.27%/0.29%
bienst1	15.11		15.49		14.98		14.99		14.99		15.25		46.75
	431.12		850.79		331.77		475.22		463.64		503.64		1.20%/1.20%
bienst2	15.28		15.35		15.35		15.28		15.28		15.27		54.6
	3586.40		3599.01		3566.95		3591.94		1194.88		1195.43		0.18%/0.18%
blp-ar98	6086.37		6086.76		6087.58		6087.57		6088.08		6088.14		6217.86
	3563.80		3559.50		3563.07		3566.11		1165.68		1179.48		1.35%/0.92%
blp-ic97	3928.3		3928.38		3928.83		3928.46		3928.64		3928.81		4057.94
	3569.97		3377.47		3586.15		3400.25		1181.92		1176.14		0.41%/0.41%
blp-ic98	4376.19		4376.8		4378.04		4378.59		4381.97		4379.05		4531.39
	3560.93		3540.84		3595.65		3541.67		1142.70		1158.75		3.72%/1.55%
blp-ir98	2283.19		2283.67		2283.51		2284.5		2286.22		2287.46		2342.32
	541.68		606.95		725.03		467.55		922.14		973.49		7.22%/2.20%
core2586-950	935.94		935.95		935.95		935.95		935.95		935.95		974
	3423.24		3502.65		3587.97		3598.87		1411.34		1205.72		0.01%/0.01%
core4284-1064	1054.08		1054.08		1054.08		1054.08		1054.08		1054.08		1086
	3598.72		3599.71		3599.47		3545.67		1196.53		1209.14		0.01%/0.00%
core4872-1529	1510.91		1510.91		1510.91		1510.91		1510.91		1510.91		1568
	3305.73		3598.09		3545.04		3599.30		1196.74		1196.19		0.00%/0.00%
danoint	62.73		62.73		62.73		62.73		62.73		62.73		65.67
	3589.38		3599.02		3598.22		3594.90		1198.09		1197.23		0.02%/0.02%
dc1c	1.7582e+06		1.7575e+06		1.7575e+06		1.7575e+06		1.7575e+06		1.7575e+06		1.8478e+06
	3591.28		3591.33		3591.63		3432.11		1192.99		1181.32		-0.84%/0.84%
dc1l	1.7445e+06		1.7445e+06		1.7445e+06		1.7446e+06		1.7446e+06		1.7445e+06		1.8517e+06
	3588.70		3588.79		3588.84		3587.28		1194.80		1195.38		0.02%/0.02%
dolom1	6.5563e+06		6.5563e+06		6.5563e+06		6.5563e+06		6.5563e+06		6.5563e+06		1.49e+08
	3588.23		3586.81		3587.44		3585.06		1189.61		1190.51		0.00%/0.00%
ds	59.54		59.56		59.49		59.59		59.59		59.55		447.01
	3584.19		3582.82		3585.46		3584.17		1194.23		1212.81		0.01%/0.01%
fast0507	172.15		172.17		172.18		172.18		172.18		172.18		174
	2340.49		2862.75		1119.82		2467.56		1170.02		1250.19		1.95%/1.87%
gesa2-o	2.5731e+07		2.5729e+07		2.5733e+07		2.5731e+07		2.5731e+07		2.5731e+07		2.5779e+07
	8.79		6.39		5.62		6.37		6.22		15.37		4.50%/4.50%
liu	560		560		560		560		560		560		1174
	3476.32		3467.47		3463.13		3452.70		1152.16		1153.31		0.00%/0.00%

Table 4. Results over reduced test set, part II

Configuration	C0		T2N5		T5N5		T10N5		T10N1k		T15N1k		Results
	Root LP Time (s)	Root LP Time (s)	Root LP Time (s)	Root LP Time (s)	Root LP Time (s)	Root LP Time (s)	Root LP Time (s)	Root LP Time (s)	Root LP Time (s)	Root LP Time (s)	Root LP Time (s)	CGap1/CGap2	
manna81	-13229.5	-13227.5	-13226.8	-13226.1	-13221.9	-13220.8	-13164						
	3540.39	3548.56	3548.17	3524.98	1181.63	1180.20	13.22%/5.21%						
mas74	10506.2	10555.8	10558.9	10535.5	10535.5	10538.2	11801.2						
	2548.22	3203.12	2437.62	2299.57	1185.31	1180.43	4.07%/4.07%						
mas76	38908	38925.2	38934.3	38911.1	38911.1	38911.8	40005.1						
	241.62	189.35	175.21	325.39	319.60	210.48	2.40%/2.40%						
misc07	1425	1425	1425	1425	1425	1425	2810						
	23.53	38.43	28.10	44.54	61.50	21.00	0.00%/0.00%						
mkc1	-611.85	-611.85	-611.85	-611.85	-611.85	-611.85	-607.21						
	19.25	20.48	39.64	99.49	89.70	58.91	0.00%/0.00%						
mkc	-582.39	-582.39	-582.39	-582.39	-579.15	-581.79	-563.23						
	3547.53	3550.57	3562.28	3567.50	1172.77	1181.56	16.91%/0.00%						
momentum1	96250.1	96250.1	96250.1	96250.1	96250.1	96250.1	109143						
	3592.25	3590.55	3594.12	3593.23	1193.81	1182.12	0.00%/0.00%						
momentum2	10702.1	10702.1	10705.4	10702.6	10702.6	10702.2	12314.2						
	3591.55	738.17	3288.60	3593.40	1196.20	749.59	0.20%/0.20%						
msc98-ip	1.9702e+07	1.9702e+07	1.9702e+07	1.9702e+07	1.9702e+07	1.9702e+07	1.98e+07						
	3583.89	3549.60	3593.76	3586.20	1196.60	1195.54	0.00%/0.00%						
mzzv11	-22066.1	-22067.3	-22066.2	-22063.3	-22061.5	-22067	-21718						
	548.37	471.03	819.70	541.25	491.16	551.90	1.32%/0.81%						
mzzv42z	-20830.7	-20826.7	-20789	-20787.2	-20787.2	-20822.6	-20540						
	130.53	140.58	155.44	157.69	161.51	226.87	14.96%/14.96%						
neos10	-1187.33	-1185.79	-1185.77	-1185.12	-1184.66	-1185.34	-1135						
	23.15	22.85	21.30	29.77	26.81	61.57	5.10%/4.23%						
neos11	6	6	6	6	6	6	9						
	446.38	405.21	192.80	216.16	231.75	203.14	0.00%/0.00%						
neos12	9.51	9.51	9.51	9.51	9.51	9.51	13						
	644.55	971.43	1111.82	883.34	1099.59	656.93	-0.01%/-0.01%						
neos13	-112.97	-112.75	-107.98	-109.42	-112.4	-112.63	-95.47						
	3596.18	3596.89	3696.19	3596.15	1190.17	1197.57	28.51%/28.51%						
neos14	66464.1	66387.1	66381.6	66439.7	66439.7	66536.2	74333.3						
	1555.23	1181.56	1426.66	1037.64	1007.33	1193.10	0.92%/-0.31%						
neos15	70411.4	70172.5	70409	70418.4	70418.4	70415.8	80835						
	3591.99	3597.23	3595.22	3588.92	1197.49	1198.19	0.07%/0.07%						
neos17	0.03	0.03	0.03	0.02	0.02	0.02	0.15						
	3057.75	315.74	776.98	514.16	517.12	1192.86	0.13%/0.13%						
neos18	13	13	13	13	13	13	16						
	325.23	3575.88	2547.07	3572.41	1183.89	1051.34	0.00%/0.00%						
neos20	-470.8	-470.8	-470.8	-470.8	-470.8	-470.8	-434						
	186.19	61.84	84.35	134.95	124.60	124.43	0.00%/0.00%						
neos21	2.72	2.74	2.73	2.73	2.74	2.75	7						
	91.88	107.15	89.11	108.62	107.98	114.33	0.71%/0.45%						
neos22	777536	777786	777676	777702	777702	777821	779715						
	66.65	52.88	49.44	36.72	38.65	62.01	13.09%/11.51%						
neos23	63.81	63.81	63.81	63.82	64.16	63.82	137						
	653.86	1958.24	2709.85	813.83	1195.49	525.27	0.48%/0.02%						
neos2	-4069.79	-4056.09	-4039.14	-4070.69	-4028.29	-3927.78	454.86						
	14.81	15.08	13.36	16.30	19.03	52.29	3.14%/0.68%						
neos3	-5664.36	-5630.32	-5655.1	-5643.04	-5643.04	-5731.26	368.84						
	54.28	53.25	54.06	69.07	63.54	80.71	0.56%/0.56%						
neos5	13.33	13.32	13.33	13.33	13.33	13.33	15						
	3197.48	3579.64	3580.21	3367.37	1193.82	1194.55	0.00%/0.00%						
neos7	692631	693168	693268	692631	692631	692532	721934						
	137.84	59.85	59.72	51.88	51.01	1091.68	2.17%/2.17%						
neos9	793.25	792.25	793.5	791.75	793.14	791.75	798						
	3587.56	3586.40	3585.06	3586.00	1191.12	1190.05	5.26%/5.26%						
nsrand-ixp	50181.8	50183.1	50184.8	50184.2	50187.8	50186.2	51200						
	3588.37	3575.79	3583.57	3609.45	1185.31	1198.76	0.58%/0.30%						

**Table 5.** Results over reduced test set, part III

Configuration	C0		T2N5		T5N5		T10N5		T10N1k		T15N1k		Results	
	Root LP	Time (s)	Root LP	Time (s)	Root LP	Time (s)	Root LP	Time (s)	Root LP	Time (s)	Root LP	Time (s)	CGap1/CGap2	Best Sol.
nsrand_jpx	50181.8		50183.1		50184.8		50184.2		50187.8		50186.2			51200
	3579.66		3583.67		3579.68		3571.66		1193.02		1203.12		0.58%/0.30%	
nug08	204.28		204.31		204.33		204.33		204.37		204.37			214
	47.44		63.14		73.89		90.37		126.99		185.04		0.92%/0.54%	
nw04	16380.3		16771.3		16779.9		16781.3		16792.6		16787.9			16862
	58.00		46.19		51.81		52.10		112.66		907.47		85.59%/83.25%	
opt1217	-20		-19		-19		-19		-19		-19			-16
	3553.84		3542.62		3544.18		3530.90		1193.49		1190.35		25.00%/25.00%	
pk1	0		0		0		0		0		0			11
	153.16		143.84		112.56		162.38		168.46		195.87		0.00%/0.00%	
profold	-41.09		-39.92		-41.09		-41.09		-39.42		-39.78			-31
	3599.38		3599.40		3599.40		3598.46		1199.45		1199.71		16.53%/11.63%	
qap10	333.5		333.5		333.48		333.51		333.52		333.51			340
	395.29		951.54		399.66		450.92		486.25		507.83		0.34%/0.23%	
qiu	-923.04		-926.83		-926.83		-923.04		-923.04		-923.04			-132.87
	77.25		92.21		95.20		77.66		74.46		68.85		0.00%/0.00%	
rail507	172.15		172.17		172.17		172.17		172.18		172.18			174
	3570.21		3320.39		3564.49		3559.79		1160.16		1255.95		1.82%/1.46%	
ran14x18_l	3362.27		3363.59		3363.68		3363.22		3363.22		3361.99			3735
	3588.84		3587.58		3587.68		3589.10		1195.36		1195.84		0.38%/0.38%	
roll3000	12243.9		12257.5		12257.8		12259.4		12259.4		12260.1			12890
	3592.85		3596.57		3595.23		3592.53		1194.49		1197.01		2.52%/2.41%	
rout	985.46		985.53		985.53		985.46		985.46		985.56			1077.56
	219.81		292.25		195.17		126.08		122.64		81.27		0.11%/0.09%	
seymour1	405.86		405.9		405.92		405.93		405.97		405.93			410.76
	2186.19		2455.05		2762.13		2433.69		1200.26		1203.71		2.22%/1.33%	
seymour	407.17		407.2		407.63		407.63		408.2		408.17			423
	3582.73		3590.80		3590.46		3590.24		1195.45		1191.93		6.50%/2.95%	
sienal	1.0163e+07		1.0163e+07		1.0163e+07		1.0163e+07		1.0163e+07		1.0163e+07			1.58e+08
	3592.54		3593.03		3592.21		3593.56		1197.19		1199.08		0.00%/0.00%	
sp97ar	6.5388e+08		6.5391e+08		6.5391e+08		6.5391e+08		6.5391e+08		6.5391e+08			6.64e+08
	3591.66		3568.81		3588.03		3568.96		1170.85		1267.79		0.36%/0.32%	
sp97ic	4.2211e+08		4.2217e+08		4.2217e+08		4.2218e+08		4.2223e+08		4.2219e+08			4.3e+08
	3553.65		3554.51		3551.67		3542.83		1136.89		1190.07		1.48%/0.79%	
sp98ar	5.2499e+08		5.2504e+08		5.2508e+08		5.2509e+08		5.2511e+08		5.2509e+08			5.3e+08
	3549.89		3556.06		3562.27		3558.49		1184.27		1175.80		2.57%/2.16%	
sp98ic	4.4430e+08		4.4445e+08		4.4443e+08		4.4445e+08		4.4448e+08		4.4444e+08			4.5141e+08
	3550.63		3552.86		3563.69		3580.06		1190.08		1174.74		2.53%/2.16%	
stein45	22		22		22		22		22		22			30
	24.73		26.15		26.43		30.23		35.56		36.11		0.00%/0.00%	
swath1	338.68		339.99		339.36		340.32		340.54		340.64			379.07
	41.33		37.34		86.34		124.20		309.76		1195.45		4.85%/4.07%	
swath2	343.09		343.89		344.07		344.13		344.27		344.91			385.2
	195.24		104.00		527.60		40.76		195.96		458.98		4.32%/2.47%	
swath3	343.09		344.09		344.16		344.13		344.29		345.07			397.76
	783.86		535.92		789.25		186.25		393.72		866.58		3.62%/1.96%	
swath	373.88		374.41		375.53		375.21		379.4		376.08			467.41
	3494.23		3533.15		3513.14		3535.65		1141.32		1199.99		5.90%/1.77%	
timtab1	443780		438838		485907		446072		446072		462240			764772
	3567.05		3587.46		3589.34		3570.40		1196.11		1193.77		13.12%/13.12%	
timtab2	563628		560505		566589		558124		558124		575232			1.1111e+06
	3583.77		3593.85		3592.81		3583.98		1194.40		1194.57		2.12%/0.54%	
tr12-30	129675		129675		129728		129762		129762		129894			130596
	3593.70		3596.10		3586.60		3544.45		1198.44		1197.57		23.72%/9.38%	
trento1	5.1830e+06		5.1830e+06		5.1830e+06		5.1830e+06		5.1830e+06		5.1830e+06			5.2874e+06
	3582.16		3590.92		3590.58		3581.38		1194.41		1195.11		0.02%/0.00%	
UMTS	2.9137e+07		2.9137e+07		2.9140e+07		2.9142e+07		2.9142e+07		2.9141e+07			3.01e+07
	3554.64		3581.09		3588.94		3576.85		1194.68		1193.78		0.47%/0.47%	

all configurations, and then the maximum of the same quantity over the T\*N5 configurations.

Figure 1 summarizes the best speed-up factor over CPLEX defaults obtained over all instances (31) that finished to optimality on all six configurations, the geometric average speed-up was 31%. Also, looking at problems where at least one configuration had finished to optimality, CPLEX was faster by at least 5% in 10 cases, while in 16, 16, 14, 11, 9 cases configurations T2N5, T5N5, T10N5, T10N1k, T15N1k, where faster by at least 5% respectively. Figure 2 summarizes the best closed  $GAP_{LP}$  over CPLEX default (1) and the best closed  $GAP_{LP}$  over CPLEX when we limit the code to add up-to five cuts at the root node (2). The number of cases where each configuration had the best root LP value where 18, 19, 27, 22, 43, 38, for C0, T2N5, T5N5, T10N5, T10N1k and T15N1k respectively.

Figure 3 shows the number of cuts added for each configuration. Note that for the T\*N1k configurations, more than 80% of the instances required less than 40 cuts. On the other hand, it seems that the more tableau rows we use to generate cuts, the less number of total cuts our procedure needs. Figure 4 shows the closed  $GAP_{LP}$  for each configuration; where negative values (i.e. the procedure performs worse than CPLEX) are displayed in the left part of the graph. Again from this figure it seems clear that there are advantages to considering more than one tableau row at the same time; in fact, the results for two tableau rows are consistently poorer than configurations that use more tableau rows in the cutting procedure.

## 5 Final Remarks and Conclusions

We have shown that even simple subclasses of inequalities derived from the infinite relaxation can have an important impact both on overall branch and cut performance, and on the  $GAP_{LP}$  closed at the root node. These results point towards both trying to identify important classes of inequalities for  $R_f$  for higher dimensions, and to find good computational implementation choices for them.

Although the implementation is numerically conservative, still, there are some numerical issues when cuts are used within the branch and bound tree. There are also instances where the cuts generated tend to be parallel to previous ones, causing again numerical issues.

There are many possibilities to explore, like adding more than one cut in every iteration, choosing different sets of tableau to work on, choosing different ground sets  $B_a$ , and testing the impact of inequalities derived from simplex-like ground sets.

## References

1. Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. *Op. Res. Letters* 33, 42–54 (2005)
2. Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. *Operations Research Letters* 34(4), 361–372 (2006)

3. Andersen, K., Louveaux, Q., Weismantel, R., Wolsey, L.A.: Inequalities from two rows of a simplex tableau. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 1–15. Springer, Heidelberg (2007)
4. Bixby, R.E., Boyd, E.A., Indovina, R.R.: MIPLIB: A test set of mixed integer programming problems. *SIAM News* 25(16) (1992)
5. Bixby, R.E., Felon, M., Gu, Z., Rothberg, E., Wunderling, R.: MIP: Theory and practice - closing the gap. In: Proceedings of the 19th IFIP TC7 Conference on System Modelling and Optimization, Deventer, The Netherlands, pp. 19–50. Kluwer, B.V., Dordrecht (2000)
6. Cook, W., Kannan, R., Schrijver, A.: Chvátal closures for mixed integer programming problems. *Math. Prog.* 47, 155–174 (1990)
7. Cornuéjols, G., Borozan, V.: Minimal valid inequalities for integer constraints. In: George Nemhauser Symposium, Atlanta (July 2007)
8. Cornuéjols, G., Margot, F.: On the facets of mixed integer programs with two integer variables and two constraints. *Tepper Working Paper 2007 E-35* (2007)
9. Gomory, R.E.: An algorithm for the mixed integer problem. Technical Report RM-2597, RAND Corporation (1960)
10. Gomory, R.E.: Corner polyhedra and two-equation cutting planes. In: George Nemhauser Symposium, Atlanta (July 2007)
11. Gomory, R.E., Johnson, E.L.: Some continuous functions related to corner polyhedra, part I. *Math. Prog.* 3, 23–85 (1972)
12. ILOG CPLEX Division, Incline Village, Nevada, 89451, USA. CPLEX 10.2 Reference Manual (2007)
13. Knuth, D.E.: *The Art of Computer Programming*. Ch. 7.2.1.1, 1st edn., vol. 4. Addison-Wesley, Reading (2005)
14. Linderoth, J.T., Savelsbergh, M.W.P.: A computational study of search strategies for mixed integer programming. *INFORMS J. on Computing* 11, 173–187 (1999)
15. Lovász, L.: Geometry of numbers and integer programming. In: Iri, M., Tanabe, K. (eds.) *Mathematical Programming: Recent Developments and Applications*, pp. 177–210. Springer, Heidelberg (1989)

# Constraint Orbital Branching

James Ostrowski<sup>1</sup>, Jeff Linderth<sup>2</sup>, Fabrizio Rossi<sup>3</sup>, and Stefano Smriglio<sup>3</sup>

<sup>1</sup> Department of Industrial and Systems Engineering, Lehigh University,  
200 W. Packer Ave., Bethlehem, PA 18015, USA  
jao204@lehigh.edu

<sup>2</sup> Department of Industrial and Systems Engineering, University of Wisconsin-Madison,  
1513 University Avenue, Madison, WI 53706, USA  
linderoth@wisc.edu

<sup>3</sup> Dipartimento di Informatica, Università di L'Aquila,  
Via Vetoio I-67010 Coppito (AQ), Italy  
{rossi, smriglio}@di.univaq.it

**Abstract.** Orbital branching is a method for branching on variables in integer programming that reduces the likelihood of evaluating redundant, isomorphic nodes in the branch-and-bound procedure. In this work, the orbital branching methodology is extended so that the branching disjunction can be based on an arbitrary constraint. Many important families of integer programs are structured such that small instances from the family are embedded in larger instances. This structural information can be exploited to define a group of strong constraints on which to base the orbital branching disjunction. The symmetric nature of the problems is further exploited by enumerating non-isomorphic solutions to instances of the small family and using these solutions to create a collection of typically easy-to-solve integer programs. The solution of each integer program in the collection is equivalent to solving the original large instance. The effectiveness of this methodology is demonstrated by computing the optimal incidence width of Steiner Triple Systems and minimum cardinality covering designs.

## 1 Introduction

Symmetry has long been considered an obstacle to solving integer programs. Recently, there has been significant work on combating symmetry in integer programs. A technique used by a variety of authors is to add inequalities that exclude symmetric feasible solutions [1,2,3]. Kaibel and Pfetsch [4] formalize many of these arguments by defining and studying the properties of a polyhedron known as an *orbitope*, the convex hull of lexicographically maximal solutions with respect to a symmetry group. Kaibel, Peinhardt, and Pfetsch [5] then use the properties of orbitopes to remove symmetry in partitioning problems. Another technique for combating symmetry is to recognize pairs of nodes of the enumeration tree that will result in symmetric feasible solutions. One of the two nodes may safely be pruned without excluding all optimal solutions from the search. This *isomorphism-free backtracking* procedure has long been used in the combinatorics community [6,7,8], and was introduced in the integer programming community with the name *isomorphism pruning* by Margot [9]. Ostrowski *et al.* [10] introduce a technique related to isomorphism pruning, called *orbital branching*. The fundamental idea behind orbital branching is to select a branching variable that is

equivalent to other variables with respect to the symmetry remaining in the problem. In this work, we extend the work [10] to the case of branching on disjunctions formed by inequalities—*constraint orbital branching*.

Exploiting the symmetry in the problem when branching on more general disjunctions of this form can often be significantly strengthened by exploiting certain types of embedded subproblem structure. Specifically, if the disjunction on which the branching is based is such that relatively few non-isomorphic feasible solutions may satisfy one side of the disjunction, then portions of potential feasible solutions may be enumerated. The original problem instance is then partitioned into smaller, more tractable problem instances. As an added benefit, the smaller instances can then easily be solved in parallel. A similar technique has been recently employed in an ad-hoc fashion by Linderoth, Margot, and Thain [11] in a continuing effort to solve an integer programming formulation for the football pool problem. This work poses a general framework for solving difficult, symmetric integer programs in this fashion.

The power of the constraint orbital branching is demonstrated by solving to optimality for the first time a well-known integer program to compute the incidence width of a Steiner Triple System with 135 elements. Previously, the largest instance in this family that was solved contained 81 elements [12]. The generality of the constraint orbital branching procedure is further illustrated by an application to the construction of minimum cardinality covering designs. In this case, the previously best known bounds from the literature are easily reproduced.

The remainder of this section contains some mathematical preliminaries, and the subsequent paper is divided into four sections. In Section 2 the constraint orbital branching method is presented and proved to be a valid branching methodology. Section 3 discusses properties of good disjunctions for the constraint orbital branching method. Section 4 describes our computational experience with the constraint orbital branching method, and conclusions are given in Section 5.

## 1.1 Preliminaries

The primary focus of this work is on set covering problems of the form

$$\min_{x \in \mathcal{F}} \{e^T x\}, \text{ with } \mathcal{F} \stackrel{\text{def}}{=} \{x \in \{0, 1\}^n \mid Ax \geq e\}, \quad (1)$$

where  $A \in \{0, 1\}^{m \times n}$  and  $e$  is a vector of ones of conformal size. The restriction of our work to set covering problems is mainly for notation convenience, but also of practical significance, since many important set covering problems contain a great deal of symmetry.

Before describing constraint orbital branching, we first define some notation. Let  $\Pi^n$  be the set of all permutations of  $I^n = \{1, \dots, n\}$ , so that  $\Pi^n$  is the symmetric group of  $I^n$ . For a vector  $\lambda \in \mathbb{R}^n$ , the permutation  $\pi \in \Pi^n$  acts on  $\lambda$  by permuting its coordinates, a transformation that we denote as

$$\pi(\lambda) = (\lambda_{\pi_1}, \lambda_{\pi_2}, \dots, \lambda_{\pi_n}).$$

Throughout this paper we display permutations in *cycle notation*. The expression  $(a_1, a_2, \dots, a_k)$  denotes a cycle which sends  $a_i$  to  $a_{i+1}$  for  $i = 1, \dots, k - 1$  and sends



$a_k$  to  $a_1$ . Some permutations can be written as a product of cycles. For example, the expression  $(a_1, a_2)(a_3)$  denotes a permutation which sends  $a_1$  to  $a_2$ ,  $a_2$  to  $a_1$ , and  $a_3$  to itself. We will omit 1-element cycles from our display.

Since all objective function coefficients in (II) are identical, permuting the coordinates of a solution does not change its objective value, i.e.  $e^T x = e^T(\pi(x)) \forall x \in \mathcal{F}$ . The *symmetry group*  $\mathcal{G}$  of (II) is the set of permutations of the variables that maps each feasible solution onto a feasible solution of the same value. In this case,

$$\mathcal{G} \stackrel{\text{def}}{=} \{ \pi \in \Pi^n \mid \pi(x) \in \mathcal{F} \quad \forall x \in \mathcal{F} \}.$$

Typically, the symmetry group  $\mathcal{G}$  of feasible solutions is not known. However, by closely examining the structure of the problem, many of the permutations making up the group can be found, and this subgroup of the original group  $\mathcal{G}$  can be employed in its place. Specifically, given a permutation  $\pi \in \Pi^n$  and a permutation  $\sigma \in \Pi^m$ , let  $A(\pi, \sigma)$  be the matrix obtained by permuting the columns of  $A$  by  $\pi$  and the rows of  $A$  by  $\sigma$ , i.e.  $A(\pi, \sigma) = P_\sigma A P_\pi$ , where  $P_\sigma$  and  $P_\pi$  are permutation matrices. Consider the set of permutations

$$\mathcal{G}(A) \stackrel{\text{def}}{=} \{ \pi \in \Pi^n \mid \exists \sigma \in \Pi^m \text{ such that } A(\pi, \sigma) = A \}.$$

For any  $\pi \in \mathcal{G}(A)$ , if  $\hat{x} \in \mathcal{F}$ , then  $\pi(\hat{x}) \in \mathcal{F}$ , so  $\mathcal{G}(A)$  forms a subgroup of  $\mathcal{G}$ , and the group  $\mathcal{G}(A)$  is the group used in our computations. The group  $\mathcal{G}(A)$  can act on an arbitrary set of points  $\mathcal{Z}$ , but in our work, it acts on either  $\mathbb{R}^n$  or  $\{0, 1\}^n$ .

For a point  $z \in \mathcal{Z}$ , the *orbit* of  $z$  under the action of the group  $\Gamma$  is the set of all elements of  $\mathcal{Z}$  to which  $z$  can be sent by permutations in  $\Gamma$ , i.e.,

$$\text{orb}(\Gamma, z) \stackrel{\text{def}}{=} \{ z' \in \mathcal{Z} \mid \exists \pi \in \Gamma \text{ such that } z' = \pi(z) \} = \{ \pi(z) \mid \pi \in \Gamma \}.$$

The stabilizer of a set  $S \subseteq \mathcal{Z}$  in  $\Gamma$  is the set of permutations in  $\Gamma$  that send  $S$  to itself.

$$\text{stab}(S, \Gamma) = \{ \pi \in \Gamma \mid \pi(S) = S \}.$$

The stabilizer of  $S$  is a subgroup of  $\Gamma$ .

At node  $a$ , the set of feasible solutions to (II) is denoted by  $\mathcal{F}(a)$ , and the value of an optimal solution for the subtree rooted at node  $a$  is denoted by  $z^*(a)$ . Two subproblems  $a$  and  $b$  are *isomorphic* if  $x \in \mathcal{F}(a) \Rightarrow \exists \pi \in \mathcal{G}$  with  $\pi(x) \in \mathcal{F}(b)$ .

## 2 Constraint Orbital Branching

Constraint orbital branching is based on the following simple observations. If  $\lambda^T x \leq \lambda_0$  is a valid inequality for (II) and  $\pi \in \mathcal{G}$ , then  $\pi(\lambda)^T x \leq \lambda_0$  is also a valid inequality for (II). In constraint orbital branching, given an integer vector  $(\lambda, \lambda_0) \in \mathbb{Z}^{n+1}$ , we will branch on a base disjunction of the form

$$(\lambda^T x \leq \lambda_0) \vee (\lambda^T x \geq \lambda_0 + 1),$$

simultaneously considering all symmetrically equivalent forms of  $\lambda x \leq \lambda_0$ . Specifically, the branching disjunction is

$$\left( \bigvee_{\mu \in \text{orb}(\mathcal{G}, \lambda)} \mu^T x \leq \lambda_0 \right) \vee \left( \bigwedge_{\mu \in \text{orb}(\mathcal{G}, \lambda)} \mu^T x \geq \lambda_0 + 1 \right).$$

Further, by exploiting the symmetry in the problem, one need only consider one representative problem for the left portion of this disjunction. That is, either the “equivalent” form of  $\lambda x \leq \lambda_0$  holds for one of the members of  $\text{orb}(\mathcal{G}, \lambda)$ , or the inequality  $\lambda x \geq \lambda_0 + 1$  holds for all of them. This is obviously a feasible division of the search space. Theorem 1 demonstrates that for any vectors  $\mu_i, \mu_j \in \text{orb}(\mathcal{G}, \lambda)$ , the subproblem formed by adding the inequality  $\mu_i^T x \leq \mu_0$  is equivalent to the subproblem formed by adding the inequality  $\mu_j^T x \leq \mu_0$ . Therefore, we need to keep only one of these representative subproblems, pruning the  $|\text{orb}(\mathcal{G}, \lambda)| - 1$  equivalent subproblems. The orbital branching (on variables) method of [10] is a special case of constraint orbital branching for  $(\lambda, \lambda_0) = (e_k, 0)$ .

**Theorem 1.** *Let  $a$  be a generic subproblem and  $\mu_i, \mu_j \in \text{orb}(\mathcal{G}, \lambda)$ . Denote by  $b$  the subproblem formed by adding the inequality  $\mu_i^T x \leq \mu_0$  to  $a$  and by  $c$  the subproblem formed by adding the inequality  $\mu_j^T x \leq \mu_0$  to  $a$ . Then,  $z^*(b) = z^*(c)$ .*

**Proof.** Let  $x^*$  be an optimal solution of  $b$ . WLOG, we can assume that  $z^*(b) \leq z^*(c)$ . Since  $\mu_i$  and  $\mu_j$  are in the same orbit, there exists a permutation  $\sigma \in \mathcal{G}$  such that  $\sigma(\mu_i) = \mu_j$ . By definition of  $\mathcal{G}$ ,  $\sigma(x^*)$  is a feasible solution to the subproblem with objective value  $z^*(b)$ . For any permutation matrix  $P$  we have that  $P^T P = I$ . Since  $x^*$  is in  $b$ ,  $\mu_i^T x^* \leq \mu_0$ . We can rewrite this as  $\mu_i^T P_\sigma^T P_\sigma x^* \leq \mu_0$ , or  $(P_\sigma \mu_i)^T P_\sigma x^* \leq \mu_0$ . This implies that  $\mu_j^T P_\sigma x^* \leq \mu_0$ , so  $\sigma(x^*)$  is in  $c$ . This implies that  $z^*(c) \leq z^*(b)$ . By our assumption,  $z^*(c) = z^*(b)$ .  $\square$

The basic *constraint orbital branching* is formalized in Algorithm 1.

---

**Algorithm 1.** Constraint Orbital Branching

---

**Input:** Subproblem  $a$ .

**Output:** Two child subproblems  $b$  and  $c$ .

---

**Step 1.** Choose a vector of integers  $\lambda$  of size  $n$  and an integer  $\lambda_0$

**Step 2.** Compute the orbit of  $\lambda$ ,  $\mathcal{O} = \{\mu_1, \dots, \mu_p\}$ .

**Step 3.** Choose arbitrary  $\mu_k \in \mathcal{O}$ . Return subproblems  $b$  with  $\mathcal{F}(b) = \mathcal{F}(a) \cap \{x \in \{0, 1\}^n : \mu_k^T x \leq \lambda_0\}$  and  $c$  with  $\mathcal{F}(c) = \mathcal{F}(a) \cap \{x \in \{0, 1\}^n : \mu_i^T x \geq \lambda_0 + 1, i = 1, \dots, p\}$

---

As for standard branching on constraints, the critical choice in Algorithm 1 is in choosing the inequality  $(\lambda, \lambda_0)$  [13]. When dealing with symmetric problems, the embedded subproblem structure can be exploited to find strong branching disjunctions, as described in the next section.

### 3 Strong Branching Disjunctions, Subproblem Structure, and Enumeration

Many important families of symmetric integer programs are structured such that small instances from the family are embedded in larger instances. In this case the problem

often shows a block-diagonal structure with identical blocks and some linking constraints, like expressed in Figure 1

The subproblem

$$z = \min_{x \in \{0,1\}^n} \{e^T x \mid Ax \geq e\},$$

denoted by  $P$ , is often computationally manageable and can be solved to optimality in reasonable time. Constraint orbital branching allows us to exploit its optimal value  $z$ . The first step consists in choosing an index  $i \in \{1, \dots, r\}$  and enforcing the constraint  $e^T x^i \geq z$ , which, obviously, does not cut off any optimal solution of the whole problem. Then, the new constraint is used as branching disjunction by letting  $\lambda = [0_n, \dots, \lambda_i, \dots, 0_n]$ ,

$\lambda_i = e_n$  and  $\lambda_0 = z$ . The resulting child subproblems have interesting properties.

*Left Subproblem.* In the left child, the constraint  $e_n^T x^i \leq z$  is added. Since also  $e_n^T x^i \geq z$  holds, this is equivalent to  $e_n^T x^i = z$ . Therefore, the feasible sub-vectors  $x^i$  for the left subproblem coincides with the set of the solutions of  $P$  with objective value equal to  $z$ . Denoting by  $\{x_1^*, x_2^*, \dots, x_l^*\}$  the set of such (optimal) solutions, one can solve the left subproblem by dividing it into  $l$  subproblems, each associated with a solution  $x_j^*$ , for  $j = 1, \dots, l$ . Precisely, each child  $j$  is generated by fixing  $x^i = x_j^*$ . This yields two relevant benefits. First, the resulting integer programs are easier than the original. Second, these are completely independent and can be solved in parallel. Of course, this option is viable only if the number of optimal solutions of  $P$  is reasonably small. Otherwise, one can select an index  $k \neq i$  and choose  $e_n^T x^k \geq z$  as a branching disjunction. In §4.1 we show how to address this “branching or enumerating” decision for well-known difficult set covering problems.

However, a more insightful observation can lessen the number of subproblems to be evaluated as children of the left subproblem. Suppose to know a symmetry group  $\mathcal{G}(P) \subseteq \Pi^n$  with the property that any two solutions in  $P$  which are isomorphic with respect to  $\mathcal{G}(P)$  generate subproblems in the original problem which are isomorphic. If such a group exists, then one can limit the search in the left subproblem only to the children corresponding to solutions  $x_j^*$  non-isomorphic with respect to  $\mathcal{G}(P)$ .

The group  $\mathcal{G}(P)$  is created as follows. Let  $I = \{i \cdot n + 1, \dots, (i + 1)n\}$  be the column indices representing the elements of  $x^i$ . First, compute the group  $\text{stab}(I, \mathcal{G})$ . Note that this group is in  $\Pi^{r \times n}$ , but we are only interested in how each  $\pi \in \text{stab}(I, \mathcal{G})$  permutes the  $n$  elements in  $I$ . For this reason, we project  $\text{stab}(I, \mathcal{G})$  onto  $I$ . Every permutation  $\pi \in 1 \text{stab}(I, \mathcal{G})$  can be expressed as a product of two smaller permutations,  $\phi \in \Pi^I$  and  $\gamma \in \Pi^{n-I}$ , where  $\phi$  permutes the elements in  $I$  and  $\gamma$  permutes the elements not in  $I$ . We can write this as  $\pi = (\phi, \gamma)$ . The projection of  $\text{stab}(I, \mathcal{G})$  onto  $I, \mathcal{G} \downarrow_I$ , contains all  $\phi$  such that there exists a  $\gamma$  with  $(\phi, \gamma) \in \text{stab}(I, \mathcal{G})$ . Note that permutations not in  $\text{stab}(I, \mathcal{G})$  cannot be projected in this way, so it is unambiguous to describe this set as  $\mathcal{G} \downarrow_I$ .

$$\min e^T x^1 + e^T x^2 + \dots + e^T x^r$$

s.t.

$$\begin{pmatrix} A & & & & \\ & A & & & \\ & & \ddots & & \\ & & & A & \\ D_1 & D_2 & \dots & D_r & \end{pmatrix} \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^r \end{pmatrix} \geq e$$

$$x^i \in \{0, 1\}^n, i = 1, \dots, r$$

Fig. 1. Block Diagonal IP

**Theorem 2.** *The projection of  $\mathcal{G}$  onto  $I$ ,  $\mathcal{G} \downarrow_I$ , is a subset of  $\mathcal{G}(P)$ .*

**Proof.** Let  $\phi \in \mathcal{G} \downarrow_I$ . Let  $x$  be any optimal solution of  $P$ . By definition,  $x$  and  $\phi(x)$  are isomorphic with respect to  $\mathcal{G} \downarrow_I$ . Consider the subproblems formed by setting  $x^i = x$  (subproblem  $a$ ) and  $x^i = \phi(x)$  (subproblem  $b$ ). By definition, there is a  $\gamma \in \Pi^{n-I}$  with  $\pi = (\phi, \gamma) \in \mathcal{G}$ .

Let  $x^*$  be any integer feasible solution in  $a$ . By definition of permutation, we know that  $\pi(x^*)$  is feasible at the root node. Also  $\pi$  sends  $x^i$  to  $\phi(x^i)$ . Since  $b$  differs from the root node only by the constraint  $x^i = \phi(x^i)$ , we have that  $\pi(x^*)$  is in  $b$ . To conclude, any solutions to  $P$  which are isomorphic with respect to  $\mathcal{G} \downarrow_I$  will generate subproblems which are isomorphic. □

**Corollary 1.** *The left subproblem can be decomposed into a set of restricted subproblems associated with the optimal solutions to  $P$  which are non-isomorphic with respect to  $\mathcal{G} \downarrow_I$ .*

In practice, non-isomorphic optimal solutions of symmetric problems often represent a small portion of all the optimal solutions. In this cases, enumerating the left subproblem becomes computationally very efficient, as shown in the case studies of Section 4.1.

*Right Subproblem.* In the right branch, the constraints  $\mu^T x \geq \lambda_0 + 1$ , for all  $\mu \in \text{orb}(\mathcal{G}, \lambda)$ , are added. If  $|\text{orb}(\mathcal{G}, \lambda)|$  is fairly large, then the LP bound is considerably increased.

The whole branching process can be iterated at the right child. In fact, the constraint  $e_n^T x^i \geq z + 1$  can be exploited as branching disjunction. In this case all the solution to  $P$  with value  $z + 1$  should be enumerated to solve the new left branch.

*Example:* Consider the graph  $G = (V, E)$  of Figure 2 and the associated vertex cover problem

$$\min_{x \in \{0,1\}^{|V|}} \{e^T x \mid x_i + x_j \geq 1 \quad \forall (i, j) \in E\}.$$

Its optimal solution has value 10 and it is supposed to be known. The coefficient matrix  $A$  shows a block diagonal structure with three blocks, corresponding to the incidence matrices of the 5-holes induced by vertices  $\{1, \dots, 5\}$ ,  $\{6, \dots, 10\}$  and  $\{11, \dots, 15\}$  respectively. Therefore, the  $i$ -th subproblem,  $i \in \{0, 1, 2\}$ , has the form

$$P : \min x_{5i+1} + x_{5i+2} + x_{5i+3} + x_{5i+4} + x_{5i+5}$$

s.t.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{5i+1} \\ x_{5i+2} \\ x_{5i+3} \\ x_{5i+4} \\ x_{5i+5} \end{pmatrix} \geq e$$

$$x \in \{0, 1\}^5$$

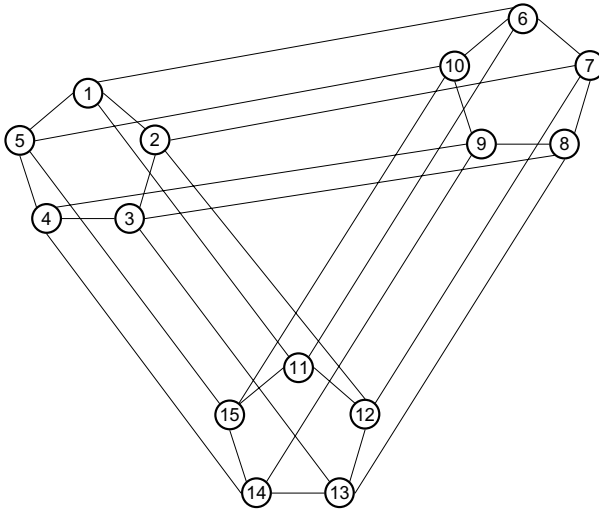


Fig. 2. Example Graph

The group  $\mathcal{G}(A)$  contains 60 permutations in  $\Pi^{15}$  and is generated by the following permutations:

$$\pi^1 = (2, 5)(3, 4)(7, 10)(8, 9)(12, 15)(13, 14)$$

$$\pi^2 = (6, 11)(7, 12)(8, 13)(9, 14)(10, 15)$$

$$\pi^3 = (1, 2)(3, 5)(6, 7)(8, 10)(11, 12)(13, 15)$$

$$\pi^4 = (1, 6)(2, 7)(3, 8)(4, 9)(5, 10).$$

$\mathcal{G}(P)$  can be created by projecting  $\mathcal{G}(A)$  on the variables of the first block (i.e.,  $x_1, \dots, x_5$ ). It consists of 10 permutations in  $\Pi^5$  which are generated by  $(2, 5)(3, 4)$ , and  $(1, 2)(3, 5)$ . The optimal solution to  $P$  has value 3 and there is only one non-isomorphic cover of size 3 (for instance,  $x_1 = 1, x_2 = 1$  and  $x_4 = 1$ ). At the root node we branch on the disjunction  $\lambda = (1, 1, 1, 1, 0, \dots, 0)$ ,  $\lambda_0 = 3$ . Then, in the left subproblem the constraint  $x_1 + x_2 + x_3 + x_4 + x_5 \leq 3$  is added, while in the right subproblem the constraints  $x_1 + x_2 + x_3 + x_4 + x_5 \geq 4, x_6 + x_7 + x_8 + x_9 + x_{10} \geq 4$  and  $x_{11} + x_{12} + x_{13} + x_{14} + x_{15} \geq 4$  are enforced.

Since  $P$  has a unique non-isomorphic optimal solution, searching the left child amounts to solve only one subproblem with  $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1$  and  $x_5 = 0$ . Its optimal value is 10 and the subproblem can be fathomed. On the right branch, the lower bound increases to 12 and also that subproblem can be fathomed.

If a classical variable-branching dichotomy is applied, it results in a much larger enumeration tree (15 subproblems vs. 3).

In the general case of unstructured problems, finding effective branching disjunctions may be difficult. Nevertheless, branching on a constraint  $(\lambda, \lambda_0)$  such that the number of non-isomorphic optimal solutions to the left subproblem is fairly small still gives good results, as shown in Section 4.3.

## 4 Case Studies

### 4.1 Steiner Triple Systems

A Steiner Triple System of order  $v$ , denoted by  $\text{STS}(v)$ , consists of a set  $S$  with  $v$  elements, and a collection  $\mathcal{B}$  of triples of  $S$  with the property that every pair of elements in  $S$  appears together in a unique triple of  $\mathcal{B}$ . Kirkman [14] showed that  $\text{STS}(v)$  exists if and only if  $v \equiv 1$  or  $3 \pmod{6}$ . A covering of a STS is a subset  $C$  of the elements of  $S$  such that  $C \cap T \neq \emptyset$  for each triple  $T \in \mathcal{B}$ . The *incidence width* of a STS is its smallest-size covering. Fulkerson, Nemhauser, and Trotter [15] suggested the following integer program to compute the incidence width of a  $\text{STS}(v)$ :

$$\min_{x \in \{0,1\}^v} \{e^T x \mid A_v x \geq 1\},$$

where  $A_v \in \{0,1\}^{|\mathcal{B}| \times v}$  is the incidence matrix of the  $\text{STS}(v)$ . The authors created instances based on STS of orders  $v \in \{9, 15, 27, 45\}$ , and posed these instances as a challenge to the integer programming community. The instance  $\text{STS}(45)$  was not solved until five years later by H. Ratliff, as reported in [16].

The instance of  $\text{STS}(27)$  was created from  $\text{STS}(9)$  and  $\text{STS}(45)$  was created from  $\text{STS}(15)$  using a “tripling” procedure described by Hall [17]. We present the construction here, since the symmetry induced by the construction is exploited by our method in order to solve larger instances in this family. For ease of notation, let the elements in  $\text{STS}(v)$  be  $\{1, 2, \dots, v\}$ , with triples  $\mathcal{B}_v$ . The elements of  $\text{STS}(3v)$  are the pairs  $\{(i, j) \mid i \in \{1, 2, \dots, v\}, j \in \{1, 2, 3\}\}$ , and its collection of triples is denoted as  $\mathcal{B}_{3v}$ . Given  $\text{STS}(v)$ , the Hall construction creates the blocks of  $\text{STS}(3v)$  in the following manner:

$$\begin{aligned} \{(a, k), (b, k), (c, k)\} &\in \mathcal{B}_{3v} \quad \forall \{a, b, c\} \in \mathcal{B}_v, \forall k \in \{1, 2, 3\}, \\ \{(i, 1), (i, 2), (i, 3)\} &\in \mathcal{B}_{3v} \quad \forall i \in \{1, \dots, v\}, \\ \{(a, \pi_1), (b, \pi_2), (c, \pi_3)\} &\in \mathcal{B}_{3v} \quad \forall \{a, b, c\} \in \mathcal{B}_v, \forall \pi \in \Pi^3. \end{aligned}$$

Feo and Resende [18] introduced two new instances  $\text{STS}(81)$  and  $\text{STS}(243)$  created using this construction.  $\text{STS}(81)$  was first solved by Mannino and Sassano [12] 12 years ago, and it remains the largest problem instance in this family to be solved.  $\text{STS}(81)$  is also easily solved by the isomorphism pruning method of [9] and the orbital branching method of [10], but neither of these methods seem capable of solving larger  $\text{STS}(v)$  instances. Karmarkar, Ramakrishnan, and Resende [19] introduced the instance  $\text{STS}(135)$  which is built by applying the tripling procedure to the  $\text{STS}(45)$  instance of [15]. Odijk and Maaren [20] have reported the best known solutions to both  $\text{STS}(135)$  and  $\text{STS}(243)$ , having values 103 and 198 respectively. Using the constraint orbital branching method, we have been able to solve  $\text{STS}(135)$  to optimality, establishing that 103 is indeed the incidence width.

The incidence matrix,  $A_{3v}$ , for an instance of  $\text{STS}(3v)$  generated by the Hall construction has the form shown in Figure 3, where  $A_v$  is the incidence matrix of  $\text{STS}(v)$  and the matrices  $D_i$  have exactly one “1” in every row. Note that  $A_{3v}$  has the block-diagonal structure discussed in Section 3, so it is a natural candidate on which to apply the constraint orbital branching methodology.

Furthermore, the symmetry group  $\Gamma$  of the instance STS( $3v$ ) created in this manner has a structure that can be exploited. Specifically for STS(135), let  $\lambda \in \mathbb{R}^{135}$  be the vector  $\lambda = (e_{45}, 0_{90})^T$  in which the first 45 components of the vector are 1, and the last 90 components are 0. It is not difficult to see that the following 12 vectors  $\mu_1, \dots, \mu_{12}$  all share an orbit with the point  $\lambda$ . (This fact can also be verified using a computational algebra package such as GAP [21]).

$$A_{3v} = \begin{bmatrix} A_v & 0 & 0 \\ 0 & A_v & 0 \\ 0 & 0 & A_v \\ I & I & I \\ D_1 & D_2 & D_3 \end{bmatrix},$$

Fig. 3. Incidence Matrix of  $A_{3v}$

	1 – 15	16 – 30	31 – 45	46 – 60	61 – 75	76 – 90	91 – 105	106 – 120	121 – 135
$\mu_1$	$e$	$e$	$e$	0	0	0	0	0	0
$\mu_2$	0	0	0	$e$	$e$	$e$	0	0	0
$\mu_3$	0	0	0	0	0	0	$e$	$e$	$e$
$\mu_4$	$e$	0	0	$e$	0	0	$e$	0	0
$\mu_5$	$e$	0	0	0	$e$	0	0	0	$e$
$\mu_6 =$	$e$	0	0	0	0	$e$	0	$e$	0
$\mu_7$	0	$e$	0	$e$	0	0	0	$e$	0
$\mu_8$	0	$e$	0	0	$e$	0	0	$e$	0
$\mu_9$	0	$e$	0	0	0	$e$	$e$	0	0
$\mu_{10}$	0	0	$e$	$e$	0	0	0	$e$	0
$\mu_{11}$	0	0	$e$	0	$e$	0	$e$	0	0
$\mu_{12}$	0	0	$e$	0	0	$e$	0	0	$e$

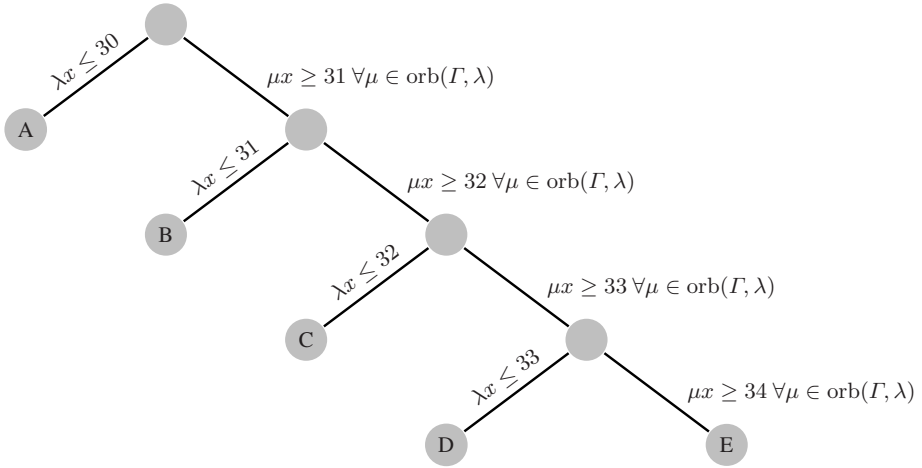
As described for the general case in Section 3, to create an effective constraint orbital branching dichotomy, we will use this orbit and also the fact that branching on the disjunction

$$(\lambda x \leq K) \vee (\mu^T x \geq K + 1) \quad \forall \mu \in \text{orb}(G, \lambda)$$

allows us to enumerate coverings for STS( $v/3$ ) in order to solve the left-branch of the dichotomy.

### 4.2 Computational Results

In this section, results of the computation proving the optimality of the cardinality 103 covering of STS(135) are presented. The optimal solution to STS(45) has value 30. Figure 4 shows the branching tree used by the constraint orbital branching method for solving STS(135). The node  $E$  in Figure 4 is pruned by bound, as the solution of the linear programming relaxation at this node is 103. A variant of the (variable) orbital branching algorithm of [10] can be used to obtain a superset of all non-isomorphic solutions to an integer program whose objective value is better than a prescribed value  $K$ . The method works in a fashion similar to that proposed in [22]. Specifically, branching and pruning operations are performed until *all* variables are fixed (nodes may not be pruned by integrality). All leaf nodes of the resulting tree are feasible solutions to the integer program whose objective value is  $\leq K$ . Using this algorithm, a superset of all non-isomorphic solutions to STS(45) of value 33 or less was enumerated. The



**Fig. 4.** Branching Tree for Solution of STS(135)

**Table 1.** Computational Statistics for Solution of STS(135)

(a) Solutions of value  $K$  for STS(45)

$(K)$	# Sol
30	2
31	246
32	9497
33	61539
71,284	

(b) Statistics for STS(135) IP Computations

$K$	Total CPU Time (sec)	Simplex Iterations	Nodes
30	538.01	2,501,377	164,720
31	90790.94	255,251,657	13,560,519
32	2918630.95	8,375,501,861	306,945,725
33	6243966.98	25,321,634,244	718,899,460
$9.16 \times 10^6$		$3.36 \times 10^{10}$	$1.04 \times 10^9$

enumeration required 10 CPU hours on a 1.8GHz AMD Opteron Processor and resulted in 71,284 solutions. The number of solutions for each value of  $K$  is shown in Table 1(a).

For each of the 71,284 enumerated solutions to STS(45), the first 45 variables of the STS(135) integer programming instance for that particular node were fixed. For example, the node  $B$  contains the inequalities  $\mu x \geq 31 \forall \mu \in \text{orb}(\Gamma, \lambda)$ , and the bound of the linear programming relaxation is 93. In order to establish the optimality of the covering of cardinality 103 for STS(135), each of these 71,284 90-variable integer programs must be solved to establish that no solution of value smaller than 103 exists. The integer programs are completely independent, so it is natural to consider solving them on a distributed computing platform. The instances were solved on a collection of over 800 computers running the Windows operating system at Lehigh University. The computational grid was created using the Condor high-throughput computing



software [23], so the computations were run on processors that would have otherwise been idle. The commercial package CPLEX (v10.2) was used to solve all the instances, and an initial upper bound of value 103.1 was provided to CPLEX as part of the input to all instances. Table 1(b) shows the aggregated statistics for the computation. The total CPU time required to solve all 71,284 instances was roughly 106 CPU days, and the wall clock time required was less than two days. The best solution found during the search had value 103<sup>1</sup>, thus establishing that the incidence-width of STS(135) is 103.

### 4.3 Covering Designs

A  $(v, k, t)$ -covering design is a family of subsets of size  $k$ , chosen from a ground set  $V$  of cardinality  $|V| = v$ , such that every subset of size  $t$  chosen from  $V$  is contained in one of the members of the family of subsets of size  $k$ . The number of members in the family of  $k$ -subsets is the covering design's size. The covering number  $C(v, k, t)$  is the minimum size of such a covering. Let  $\mathcal{K}$  be the collection of all  $k$ -sets of  $V$ , and let  $\mathcal{T}$  be the collection of all  $t$ -sets of  $V$ . An integer program to compute a  $(v, k, t)$ -covering design can be written as

$$\min_{x \in \{0,1\}^{|\mathcal{K}|}} \{e^T x \mid Bx \geq e\}, \tag{2}$$

where  $B \in \{0, 1\}^{|\mathcal{T}| \times |\mathcal{K}|}$  has element  $b_{ij} = 1$  if and only if  $t$ -set  $i$  is contained in  $k$ -set  $j$ , and the decision variable  $x_j = 1$  if and only if the  $j$ th  $k$ -set is chosen in the covering design.

Numerous theorem exist that give bounds on the size of the covering number  $C(v, k, t)$ . An important theorem that we need to generate a useful branching disjunction for the constraint orbital branching method is due to Schönheim [24]. For some subset of the ground set elements  $U \subseteq V$ , let  $\mathcal{K}(U)$  be the collection of all the  $k$ -sets of  $V$  that contain  $U$ . Margot [25] shows that the following inequality, which he calls a Schönheim inequality, is valid, provided that  $|U| = u$  is such that  $1 \leq u \leq t - 1$ :

$$\sum_{i \in \mathcal{K}(U)} x_i \geq C(v - u, k - u, t - u). \tag{3}$$

The Schönheim inequalities substantially increase the value of the linear programming relaxation of (2).

A second important observation is that the symmetry group  $G$  for (2) is such the characteristic vectors of all  $u$ -sets belong to the same orbit: if  $|U'| = |U|$ , then  $\chi_{\mathcal{K}(U')} \in \text{orb}(G, \chi_{\mathcal{K}(U)})$ . These two observations taken together indicate that the Schönheim inequalities (3) may be a good candidate for constraint orbital branching. On the left branch, the constraint

$$\sum_{i \in \mathcal{K}(U)} x_i \leq C(v - u, k - u, t - u)$$

is enforced. To solve this node, all non-isomorphic solutions to the  $(v - u, k - u, t - u)$ -covering design problem may be enumerated. For each of these solutions, an integer

<sup>1</sup> In fact, two solutions of value 103 were found, but they were isomorphic.

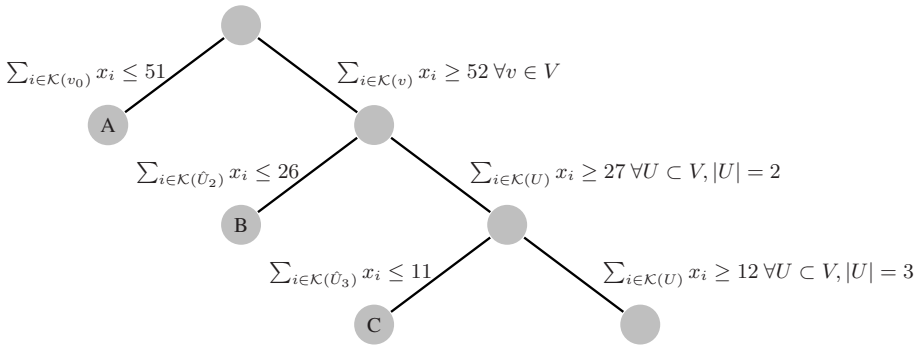


Fig. 5. Branching Tree for  $C(11, 6, 5)$

program in which the corresponding variables in the  $(v, k, t)$ -covering design problem are fixed may be solved. On the right branch of the constraint-orbital branching method, the constraints

$$\sum_{i \in \mathcal{K}(U')} x_i \geq C(v - u, k - u, t - u) + 1 \quad \forall U' \in \text{orb}(G, \chi_{\mathcal{K}(U)})$$

may be imposed. These inequalities can significantly improve the linear programming relaxation.

#### 4.4 Computational Results

We will demonstrate the applicability of constraint orbital branching using the Schönheim inequalities by an application to the  $(11, 6, 5)$ -covering design problem. Nurmela and Östergård [26] report an upper bound of  $C(11, 6, 5) \leq 100$ , and Applegate, Rains, and Aloane [27] were able to show that  $C(11, 6, 5) \geq 96$ . Using the constraint orbital branching technique, we are also easily able to obtain the bound  $C(11, 6, 5) \geq 96$ , and ongoing computations are aimed at further sharpening the bound. The covering design numbers  $C(10, 5, 4) = 51$ ,  $C(9, 4, 3) = 25$ , and  $C(8, 3, 2) = 11$  are all known [28], and this knowledge is used in the branching scheme.

The branching tree used for the  $(11, 6, 5)$ -covering design computations is shown in Figure 5. In the figure, node  $D$  is pruned by bound, as the value of its linear programming relaxation is  $> 100$ . The nodes  $A$ ,  $B$ , and  $C$  will be solved by enumerating solutions to a  $(v, k, t)$ -covering design problem of appropriate size. For node  $A$ ,  $(10, 5, 4)$ -covering designs of size 51 are enumerated; for node  $B$ ,  $(9, 4, 3)$ -covering designs of size  $\leq 26$  are enumerated; and for node  $C$ ,  $(8, 3, 2)$ -covering designs of size  $\leq 11$  are enumerated. Table 2 shows the number of solutions at each node, as well as the value of the linear programming relaxation  $z(\rho)$  of the parent node. The size 51  $(10, 5, 4)$ -covering designs are taken from the paper [29], and the other covering designs are enumerated using the variant of the orbital branching method outlined in Section 4.2.

Since the value of the linear programming relaxation of the parent of node  $B$  is 95.33, if none of the 40 integer programs created by fixing the size 51 (10, 5, 4)-covering design solutions at node  $A$  of Figure 5 has a solution of value 95, then immediately, a lower bound of  $C(11, 6, 5) \geq 96$  is proved. The computation to improve the lower bound for each of the 40 IPs to 95.1 required only 8789 nodes and 10757.5 CPU seconds on a single 2.60GHz Intel Pentium 4 CPU. More extensive computations are currently underway on a Condor-provided computational grid to further improve this bound.

It is interesting to note that an attempt to improve the lower bound of  $C(11, 6, 5)$  by a straightforward application of the variable orbital branching method of [10] was unable to improve the bound higher than 94, even after running several days and eventually exhausting a 2GB memory limit. An exhaustive comparison with variable orbital branching will be reported in a journal version of the paper. However, the results on specific classes of problems show that the generality of constraint orbital branching does appear to be useful to solve larger symmetric problems.

**Table 2.** Node Characteristics

Node	# Sol	$z(\rho)$
$A$	40	93.5
$B$	782,238	95.33
$C$	11	99

## 5 Conclusions

In this work, we generalized a previous work for branching on orbits of variables (orbital branching) to branching on orbits of constraints (constraint orbital branching). Constraint orbital branching can be especially powerful if the problem structure is exploited to identify a strong constraint on which to base the disjunction and by enumerating all partial solutions that might satisfy the constraint. Using this methodology, we are for the first time able to establish the optimality of the cardinality 103 covering for STS(135).

## Acknowledgment

The authors would like to thank François Margot for his insightful comments about this work and Helen Lindereth for hand-keying all 40 non-isomorphic solutions to  $C(10, 5, 4)$  into text files. Author Lindereth would like to acknowledge support from the US National Science Foundation (NSF) under grant DMI-0522796, by the US Department of Energy under grant DE-FG02-05ER25694, and by IBM, through the faculty partnership program. The solution of the STS135 instance could not have been achieved were it not for the generous donation of “unlimited” CPLEX licenses by Rosemary Berger and Lloyd Clarke of ILOG. The authors dedicate this paper to the memory of their good friend Lloyd.

## References

1. Macambira, E.M., Maculan, N., de Souza, C.C.: Reducing symmetry of the SONET ring assignment problem using hierarchical inequalities. Technical Report ES-636/04, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro (2004)

2. Rothberg, E.: Using cuts to remove symmetry. In: 17<sup>th</sup> International Symposium on Mathematical Programming (2000)
3. Sherali, H.D., Smith, J.C.: Improving zero-one model representations via symmetry considerations. *Management Science* 47(10), 1396–1407 (2001)
4. Kaibel, V., Pfetsch, M.: Packing and partitioning orbitopes. *Mathematical Programming* (to appear, 2007)
5. Kaibel, V., Peinhardt, M., Pfetsch, M.: Orbitopal fixing. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 74–88. Springer, Heidelberg (2007)
6. Read, R.C.: Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. *Annals of Discrete Mathematics* 2, 107–120 (1998)
7. Butler, G., Lam, W.H.: A general backtrack algorithm for the isomorphism problem of combinatorial objects. *Journal of Symbolic Computation* 1, 363–381 (1985)
8. McKay, D.: Isomorph-free exhaustive generation. *Journal of Algorithms* 26, 306–324 (1998)
9. Margot, F.: Pruning by isomorphism in branch-and-cut. *Mathematical Programming* 94, 71–90 (2002)
10. Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 104–118. Springer, Heidelberg (2007)
11. Linderoth, J., Margot, F., Thain, G.: Improving bounds on the football pool problem via symmetry reduction and high-throughput computing (submitted, 2007)
12. Mannino, C., Sassano, A.: Solving hard set covering problems. *Operations Research Letters* 18, 1–5 (1995)
13. Karamanov, M., Cornuéjols, G.: Branching on general disjunctions (submitted, 2005)
14. Kirkman, T.P.: On a problem in combinations. *Cambridge and Dublin Mathematics Journal* 2, 191–204 (1847)
15. Fulkerson, D.R., Nemhauser, G.L., Trotter, L.E.: Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triples. *Mathematical Programming Study* 2, 72–81 (1974)
16. Avis, D.: A note on some computationally difficult set covering problems. *Mathematical Programming* 8, 138–145 (1980)
17. Hall, M.: *Combinatorial Theory*. Blaisdell Company (1967)
18. Feo, T.A., Resende, G.C.: A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters* 8, 67–71 (1989)
19. Karmarkar, N., Ramakrishnan, K., Resende, M.: An interior point algorithm to solve computationally difficult set covering problems. *Mathematical Programming, Series B* 52, 597–618 (1991)
20. Odijk, M.A., van Maaren, H.: Improved solutions to the Steiner triple covering problem. *Information Processing Letters* 65(2), 67–69 (1998)
21. The GAP Group: GAP—Groups, Algorithms, and Programming, Version 4.4 (2004), <http://www.gap-system.org>
22. Danna, E., Fenelon, M., Gu, Z., Wunderling, R.: Generating multiple solutions for mixed integer programming problems. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 280–294. Springer, Heidelberg (2007)
23. Livny, M., Basney, J., Raman, R., Tannenbaum, T.: Mechanisms for high throughput computing. *SPEEDUP 11* (1997)
24. Schönheim, J.: On coverings. *Pacific Journal of Mathematics* 14, 1405–1411 (1964)
25. Margot, F.: Exploiting orbits in symmetric ILP. *Mathematical Programming, Series B* 98, 3–21 (2003)

26. Nurmela, K.J., Östergård, P.: Upper bounds for covering designs by simulated annealing. *Congressus Numerantium* 96, 93–111 (1993)
27. Applegate, D., Rains, E., Sloane, N.: On asymmetric coverings and covering numbers. *Journal of Combinatorial Designs* 11, 218–228 (2003)
28. Gordon, D., Kuperberg, G., Patashnik, O.: New constructions for covering designs. *Journal of Combinatorial Designs* 3, 269–284 (1995)
29. Margot, F.: Small covering designs by branch-and-cut. *Mathematical Programming* 94, 207–220 (2003)

# A Fast, Simpler Algorithm for the Matroid Parity Problem

James B. Orlin

MIT Sloan School of Management  
Cambridge, MA 02139  
jorlin@mit.edu

**Abstract.** Consider a matrix with  $m$  rows and  $n$  pairs of columns. The linear matroid parity problem (LMPP) is to determine a maximum number of pairs of columns that are linearly independent. We show how to solve the linear matroid parity problem as a sequence of matroid intersection problems. The algorithm runs in  $\mathcal{O}(m^3n)$ . Our algorithm is comparable to the best running time for the LMPP, and is far simpler and faster than the algorithm of Orlin and Vande Vate [10], who also solved the LMPP as a sequence of matroid intersection problems. In addition, the algorithm may be viewed naturally as an extension of the blossom algorithm for nonbipartite matchings.

**Keywords:** Matroid parity, matroid matching, matroid intersection, nonbipartite matching.

## 1 Introduction

Let  $E^* = \{e_1, e_2, \dots, e_n\}$  be a set of lines, where line  $e_k = (e_{k1}, e_{k2})$  is a pair of points in  $\mathbb{R}^m$ , where  $\mathbb{R}$  is the set of real numbers. The

(LMPP) is to find a maximum collection  $M$  of lines so that all points of  $M$  are linear independent. The non-bipartite matching problem is the special case of the LMPP in which  $e_{kj}$  is a unit vector for all  $k = 1$  to  $n$  and  $j = 1, 2$ . The LMPP also generalizes the matroid intersection problem in the case that the matroids are linear.

The matroid parity problem is the generalization of the LMPP in which each line consists of two elements of a matroid, and  $M$  is required to consist of elements that are independent with respect to the matroid. Lovász [8] and Jensen and Korte [4] independently showed that the matroid parity problem requires exponential time in the case that independence with respect to the matroid is evaluated by an oracle function.

In this paper, we give an algorithm that runs in  $\mathcal{O}(n^3m)$  time that solves the LMPP as a sequence of matroid intersection problems. Our approach substantially simplifies and speeds up the approach of Orlin and Vande Vate [10], who solved the linear matroid parity problem as a sequence of matroid intersection problems in  $\mathcal{O}(n^4m)$  time. The approach presented here may be viewed as

a generalization of Edmonds’ blossom algorithm [1]. We compare the blossom algorithm to our algorithm at several different points in our paper.

Lovász [8] developed the first polynomial time algorithm for the linear matroid parity problem. Lovász and Plummer [9] reported that its running time if implemented in a straightforward manner is  $\mathcal{O}(n^{17})$ , and it can be implemented to run in  $\mathcal{O}(n^{10})$  time.

Stallmann and Gabow [12] and Gabow and Stallmann [3] solved the matroid parity problem in  $\mathcal{O}(n^3m)$  time using some ideas from the blossom algorithm of Edmonds [1]. (Their algorithm is quite different from ours). The latter algorithm can be made faster by using fast matrix multiplication. For applications and algorithms on closely related problems see Lovász [7], Lovász and Plummer [9] and Schrijver [11]. Our algorithm achieves the same running time as that of Gabow and Stallmann [3], assuming that fast matrix multiplication is not used. (This author does not know whether fast matrix multiplication leads to a comparable speed up of the algorithm presented here).

The outline of this paper is as follows. In Section 2, we present background on the LMPP. Section 3 presents Lovász’s duality theorem for the LMPP and some elementary consequences. Section 4 provides a high level view of the algorithm. Sections 5 to 10 fill in details on the algorithm. Section 11 discusses computational issues. Section 12 provides a brief summary. Proofs of some of the main theorems are included in an appendix.

## 2 Linear Matroids, Matchings, and Pseudomatchings

For a subset  $S \subseteq \mathbb{R}^m$ , let  $r(S)$  denote the linear rank of  $S$ . We say that  $S$  is independent if  $r(S) = |S|$ . A subset  $I$  of a set  $S$  is a maximally independent subset of  $S$ . A subset  $D$  of  $S$  is a minimally dependent subset of  $S$ . We sometimes let  $\mathcal{I}$  denote the collection of independent sets. We use the notation  $S+s$  as shorthand for  $S \cup \{s\}$ .

The orthogonal complement of  $S$  is  $\overline{S} = \{s \in \mathbb{R}^m : r(S + s) = r(S)\}$ . We say that two subsets  $S$  and  $T$  of  $\mathbb{R}^m$  are orthogonal if  $r(S \cup T) = r(S) + r(T)$ . (This is not a standard use of the term “orthogonal” in linear algebra since we are not really requiring the linear spaces induced by  $S$  and  $T$  to be perpendicular to each other. We hope that it will not lead to any confusion).

For given subsets  $S, Q \subseteq \mathbb{R}^m$ , we let  $r_Q(S) = r(S \cup Q) - r(Q)$ . We say that  $r_Q(\cdot)$  is obtained from  $r(\cdot)$  by contracting  $Q$ .

Given a linear matroid parity problem defined on  $E^*$ , a matching  $M$  is a set of lines whose points are independent. We let  $|M|$  denote the number of points (not lines) in  $M$ . So,  $M$  is a matching if  $r(M) = |M|$ . (We will explain later why we keep track of the number of points rather than the number of lines).

Partitions play an important role in the linear matroid parity algorithm. Suppose that  $E \subseteq E^*$ , and  $\mathcal{P} = (E_1, \dots, E_K)$  is a partition of the lines of  $E$ . We refer to each part  $E_j$  as a part of  $\mathcal{P}$ .

A  $\mathcal{P}$ -matching is a subset  $M$  of points of  $E$  such that  $|M \cap E_j|$  is even for  $j = 1$  to  $K$ . The  $\mathcal{P}$ -matching problem is to find a  $\mathcal{P}$ -matching of maximum size.

cardinality. The  $\mathcal{P}$ -matching problem is both a generalization of the matroid parity problem and a relaxation of the matroid parity problem. If the partition  $\mathcal{P}$  is not specified, we will typically refer to  $M$  as a  $\mathcal{P}$ -matching. A matching is a special case of a pseudomatching.

**Definition 1.** Let  $E \subseteq E^*$ ,  $\omega(E)$  be the cardinality of  $E$ ,  $P$  a partition of  $E$ ,  $\omega(\mathcal{P})$  the sum of the cardinalities of the parts of  $\mathcal{P}$ , and  $\mathcal{P}$ -matching a matching  $M$  such that  $M \cap E_j$  is a matching in  $E_j$  for each  $E_j \in \mathcal{P}$ .

Earlier we defined the size of matchings in terms of points because it is a more natural way of measuring the size of pseudomatchings. We also note that the  $\mathcal{P}$ -matching problem is polynomially transformable to the LMPP. All one needs to do is replace each component  $E_j$  with  $k$  points by  $\binom{k}{2}$  lines consisting of the pairs of points of  $E_j$ . Then any matching for the transformed problem is a  $\mathcal{P}$ -matching for the original problem.

### 3 Duality for the LMPP

In this section, we describe Lovász’s duality theorem [11] for the linear matroid parity problem as well as some lemmas that are needed for our algorithm.

**Definition 2.** Let  $\mathcal{P} = (E_1, \dots, E_K)$  be a partition of  $E$ ,  $Q \subseteq \overline{E}$  a set of points,  $(\mathcal{P}, Q)$  a dual solution, and  $\mu(\mathcal{P}, Q)$  the capacity of  $(\mathcal{P}, Q)$ .

$$\mu(\mathcal{P}, Q) = 2r(Q) + 2 \sum_{i=1}^K \lfloor r_Q(E_j)/2 \rfloor.$$

The reader should note that the sizes of a matchings and the capacity of dual solutions in this paper are twice the values of corresponding terms in other papers.

**Lemma 1 (Lovász [8] weak duality).** Let  $\mathcal{P} = (E_1, \dots, E_K)$  be a partition of  $E$ ,  $Q \subseteq \overline{E}$  a set of points,  $(\mathcal{P}, Q)$  a dual solution, and  $M$  a  $\mathcal{P}$ -matching. Then  $\omega(M) \leq \mu(\mathcal{P}, Q)$ . □

**Theorem 1 (Lovász [8] strong duality).** Let  $\mathcal{P} = (E_1, \dots, E_K)$  be a partition of  $E$ ,  $Q \subseteq \overline{E}$  a set of points,  $(\mathcal{P}, Q)$  a dual solution, and  $M$  a  $\mathcal{P}$ -matching. Then  $\omega(M) = \mu(\mathcal{P}, Q)$ . □

Actually, Lovász did not consider pseudomatchings and thus did not consider  $\omega(\mathcal{P})$ . Our algorithm can be used to give an alternative proof of Lovász’s strong duality theorem. However, we will accept the Lovász’s theorem as true for our paper. The Lovász strong duality theorem immediately implies certain structural properties of the maximum cardinality matchings of any subset  $E \subseteq E^*$  as well as of the  $\mathcal{P}$ -relaxation. We state them next.



**Corollary 1.**  $\mathcal{P} = (E_1, \dots, E_K)$   $E,$   
 $(\mathcal{P}, Q)$   $E.$   $M$   
 $\mathcal{P}$ -  $E.$  ( $\mathcal{P}$ -  $1).$   $M_i = M \cap E_i$

$i = 1 \dots K.$

1.  $Q \subseteq \overline{M}.$
2.  $r_Q(E_j) \leq 1, E_j \subseteq \overline{M}, |M_j| = r_Q(E_j).$
3.  $r_Q(E_j) \leq 1, |M_j| = [r_Q(E_j) \pm 1].$   
 $(\ ) \quad |M_j| = r_Q(E_j) - 1, \quad r_Q(M_j) = 2|M_j|.$   
 $(\ ) \quad |M_j| = r_Q(E_j) + 1, \quad r_Q(M_j) = 2|M_j| - 1. \quad \square$

**Definition 3.**  $E_j$  even component odd component  
 $(\mathcal{P}, Q)$   $r_Q(E_j)$   $3a,$   $M_j$   
 $(\mathcal{P}, Q)$ -hypomatching  $E_j.$   $3b,$   $M_j$   $(\mathcal{P}, Q)$ -hyper-  
 matchings  $E_j.$  hypomatchings hyper-  
 matchings  $(\mathcal{P}, Q)$

### 4 An Overview of the Matroid Parity Algorithm

In Figure 1, we give a flowchart of the augmentation phase of the matroid parity algorithm. This phase starts with a matching and looks for a larger cardinality matching. We describe the augmentation phase in more detail in this section.

The algorithm maintains a 6-tuple  $(E, S, \mathcal{P}, Q, \mathcal{M}, \mathcal{H}).$

- $E \subseteq E^*$  is a set of “scanned lines”.
- The set  $S$  is a set of  $\mathbb{R}^m$  points, which are points in  $\mathbb{R}^m$  that may include points that are not in  $E.$  These serve a similar purpose to pseudonodes in the blossom algorithm for nonbipartite matching (see Edmonds [11]).
- $(\mathcal{P}, Q)$  is a minimum capacity dual solution of  $E$  at the beginning of the main loop of the diagram.
- $\mathcal{M}$  is a collection of maximum cardinality pseudomatchings of  $E$  at the beginning of the main loop. It contains at least one matching.
- $\mathcal{H}$  is a collection of hypermatchings. They are needed in order to convert a pseudomatching to a matching of the same cardinality, in the Procedure **Transform-Pseudomatching**, as described in Section 9.

**Initialization.** The 6-tuple at the beginning of each augmentation phase (after a larger matching is obtained) consists of a single matching  $M.$  In this case,  $\mathcal{M} = \{M\},$  and  $E = M.$  Each component of  $\mathcal{P}$  is a line of  $M.$  The sets  $S, Q,$  and  $H$  are all empty at the beginning of an augmentation phase. Initially,  $M = \emptyset.$

**Selecting an Eligible Line.** In Step 2, the algorithm seeks out an eligible line  $e$  in  $E^* \setminus E.$  We define “eligible” in Section 6 in such a way that if there are no eligible lines (see Step 3), then any matching in  $\mathcal{M}$  is a maximum pseudomatching for the LMPP defined on  $E^*,$  and the algorithm terminates. Otherwise, the algorithm continues by adding  $e$  to  $E$  and appending the singleton component  $\{e\}$  to  $\mathcal{P}.$  After this step, if  $M$  is a pseudomatching of  $\mathcal{M},$  then  $|M| = \mu(\mathcal{P}, Q) - 2.$

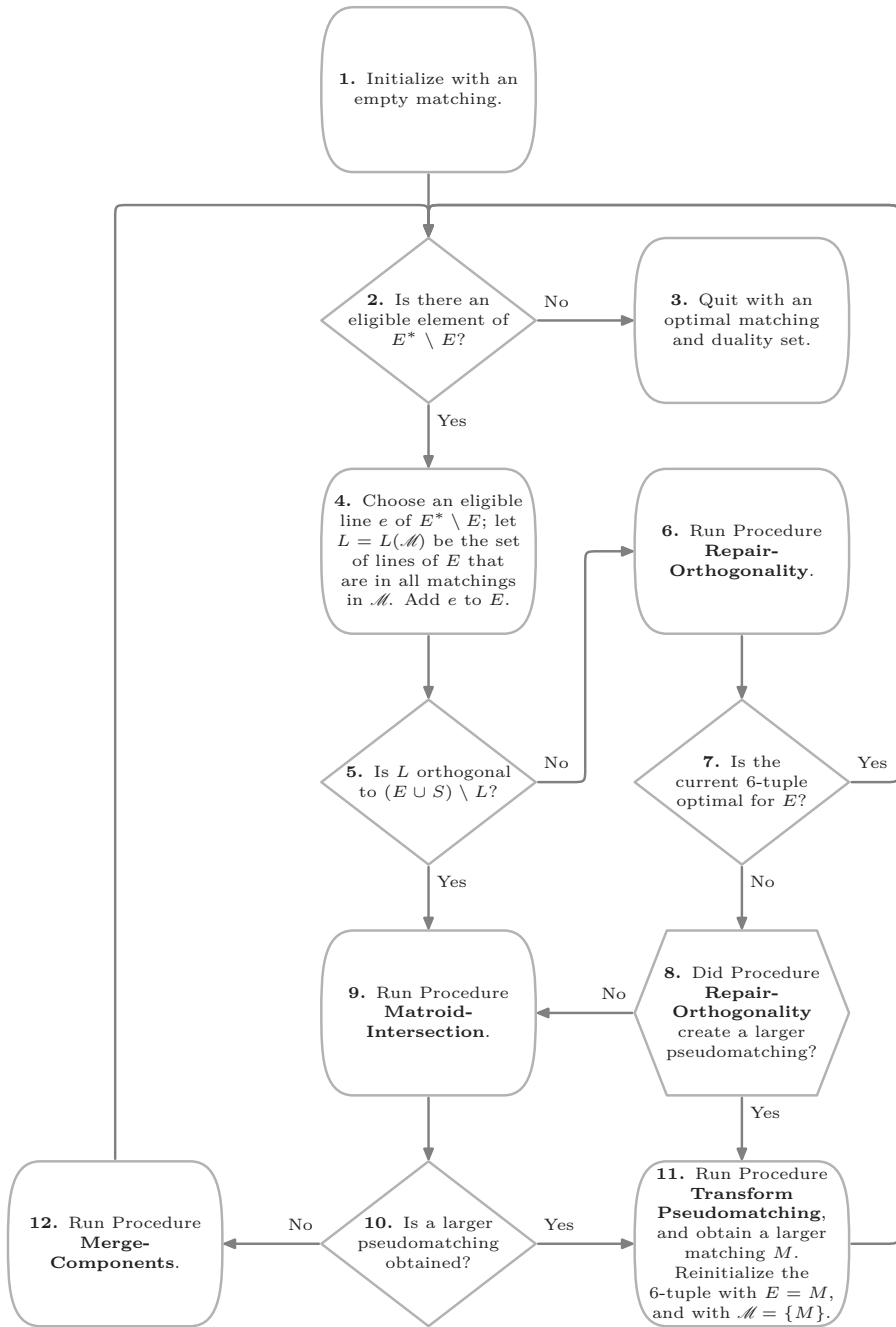


Fig. 1. A flow chart of the linear matroid parity algorithm

**Orthogonality Requirements.** The algorithm then (Step 4) determines the set  $L = L(\mathcal{M})$ , which is the collection of lines that are in every pseudomatching of  $\mathcal{M}$ . In Step 9, the algorithm needs  $L$  to be orthogonal to  $E \setminus L$ . If the two sets are not orthogonal, then Procedure **Repair-Orthogonality** (Step 6, and described in Section 7) selects a pseudomatching  $M \in \mathcal{M}$  such that  $r(M + e) = r(M) + 1$ . The procedure then adds to  $\mathcal{M}$  several new pseudomatchings of the form  $M + e - e'$  for some lines  $e' \in M$ . It also modifies  $Q$ ,  $\mathcal{M}$ , and  $L(\mathcal{M})$  so that the orthogonality conditions are once again satisfied.

**Matroid Intersection.** The algorithm then solves a matroid intersection problem (Step 9, as described in Section 8). Procedure **Matroid-Intersection** accomplishes one of the following: (1) it finds a larger pseudomatching, or (2) it creates a dual solution  $(\mathcal{P}', Q')$  for  $E$  such that for  $M \in \mathcal{M}$ ,  $|M| = \mu(\mathcal{P}', Q')$ . In case (2), the algorithm uses matroid intersection dual solutions (see, for example, Lawler [6]) in order to determine which components of  $\mathcal{P}$  to merge, thus creating a modified partition  $\mathcal{P}'$ . The algorithmic counterpart of Procedure **Matroid-Intersection** within the blossom algorithm is the identification and shrinking of blossoms.

**Transforming Pseudomatchings.** If a larger pseudomatching  $M'$  is found, the algorithm then runs **Transform Pseudomatching** (Step 11), which iteratively expands the merged components one at a time, and simultaneously obtains a maximum cardinality pseudomatching at each expansion. At the end of this procedure, the algorithm obtains a maximum cardinality matching  $M'$  for  $E$ . The augmentation phase then ends. A new augmentation phase begins with the matching  $M'$ . The subroutine **Transform Pseudomatching** has an algorithmic counterpart of expanding pseudonodes in the blossom algorithm.

**Obtaining Hypermatchings and Pseudopoints.** If the algorithm merges two or more components, it then determines  $(\mathcal{P}', Q')$ -hypermatchings for the new component and adds them to  $\mathcal{H}$ . The algorithm also adds new  $\mathcal{P}$ -matchings to  $\mathcal{M}$ , and it adds  $S$ , which are needed for subsequent matroid intersection problems. The  $\mathcal{P}$ -matchings and hypermatchings are both needed in order carry out any subsequent **Transform Pseudomatching**. The  $\mathcal{P}$ -matchings are determined directly from the matroid intersection problem. The  $(\mathcal{P}', Q')$ -hypermatchings are determined directly from a slightly modified matroid intersection problem, as described in the appendix. After the merging of components, the algorithm returns to Step 1 and begins the main loop again. Within the non-bipartite matching algorithm, the hypermatchings correspond to edges in the shrunk graph that are incident to a pseudonode.

**On Linearity.** There are two locations where the algorithm explicitly uses the fact that the points are all vectors in  $\mathbb{R}^m$  as opposed to being in a matroid. The locations are in **Repair Orthogonality** and procedure for creating pseudopoints. We will identify where linearity is assumed when we discuss the procedures in the subsequent sections.

## 5 Algorithm Invariants

We describe the algorithm in further detail in this and subsequent sections. In this section, we describe invariants that are satisfied by the algorithm at intermediate steps. We will use the notation of 6-tuples  $(E, S, \mathcal{P}, Q, \mathcal{M}, \mathcal{H})$  given in the previous section.

**Invariant 1 (Independence and Orthogonality of the Even Components).**  $L = L(\mathcal{M})$ .  $e \in L, \{e\} \cap \mathcal{P} = \emptyset$ .  $r(E \setminus L) + r(L) = r(E)$ . Invariant 1 is satisfied at Steps 1, 2, 3, 7, 8, and at the end of Steps 11, and 12.

**Invariant 2 (Primal-Dual Optimality).**  $M \in \mathcal{M}, |M| = \mu(\mathcal{P}, Q)$ .  $M \cap E = \emptyset$ . (All elements of  $\mathcal{M}$  are pseudomatchings). Invariant 2 is satisfied at Steps 1, 2, 3, and at the end of Steps 11 and 12.

**Invariant 3 (Criticality of  $Q$ ).**  $\bigcap_{M \in \mathcal{M}} \overline{M \setminus L} = \overline{Q}$ . Invariant 3 is satisfied at Steps 1, 2, 4, 5, 7, 8, 9, and at the end of Steps 11 and 12.

**Invariant 4 (Critical Points and Hypermatchings).**  $S_i \subseteq \mathbb{R}^m$ .  $E_i \cap \mathcal{P} = \emptyset$ .  $\overline{E_i} \cap \overline{Q} = \overline{H(p)}$ .  $p \in \overline{H(p)} \cap \overline{Q}$ . (More precisely,  $p \in \overline{H(p)} \cap \overline{Q(p)}$  where  $Q(p)$  is the set of critical points at the iteration at which pseudopoint  $p$  was created). Invariant 4 is satisfied at all times except at the end of Step 10. It is reestablished in Step 11.

To a great extent, the various procedures are designed to reestablish invariants. For example, Invariants 1 and 2 are both potentially unsatisfied after an eligible line is added to  $E$ . Procedure **Repair-Orthogonality** reestablishes Invariant 1. Procedures **Matroid-Intersection** followed by either **Merge-Components** or **Transform Pseudomatching** reestablishes Invariant 2.

## 6 Eligible Lines and Optimality Conditions

In this brief section, we discuss when lines are eligible, and why the lack of eligible lines means that there is a matching in  $\mathcal{M}$  that is optimum for the SMPP. We say that a line  $e \in E^* \setminus E$  is eligible with respect to  $(E, S, \mathcal{P}, Q, \mathcal{M}, \mathcal{H})$  if  $r(e + Q) \leq r(Q) + 1$  or if  $r_Q(e + E_j) = r_Q(E_j)$  for any odd component of  $\mathcal{P}$ , or if  $r_Q(e + L) = r_Q(L)$ , where  $L = L(\mathcal{M})$ . Otherwise, we say that  $e$  is ineligible.

The algorithm will terminate when there are no eligible lines. We see this in the following theorem.

**Theorem 2.** If  $(E, S, \mathcal{P}, Q, \mathcal{M}, \mathcal{H})$  is feasible and  $e \in E^* \setminus E$ , then there exists a matching  $M \in \mathcal{M}$  such that  $|M| = 1$  and  $M \cap E = \emptyset$ .

- $$E^* \quad , \quad fi$$
1.  $\mathcal{P}'$  -
  2.  $e \in E^* \setminus E, \quad r(e + Q) \leq r(Q) + 1,$
  3.  $e \in E^* \setminus E, \quad r(e + Q) = r(Q) + 2 \quad r_Q(e + E_j) =$   
 $r_Q(E_j) \quad E_j \quad \mathcal{P}', \quad e \quad E_j. \quad \square$

### 7 Restoring Orthogonality

This section presents Procedure **Repair-Orthogonality**. Suppose that  $(E, S, \mathcal{P}, Q, \mathcal{M}, \mathcal{H})$  satisfies Invariants 1 to 4, and suppose that  $e$  is an eligible element of  $E^* \setminus E$ . Let  $\mathcal{P} + \{e\}$  be shorthand for the partition obtained from  $\mathcal{P}$  by appending a singleton component  $\{e\}$ . Let  $L = L(M)$ . Since  $e$  is eligible, it follows that  $r(e + Q) = r(Q) + 2$  and  $r_Q(e + L) > r_Q(L)$ .

If  $L$  is orthogonal to  $(E + e) \setminus L$ , then Invariant 1 is satisfied by the 6-tuple  $(E + e, S, \mathcal{P} + \{e\}, Q, \mathcal{M}, \mathcal{H})$ , in which case we would skip Procedure **Repair-Orthogonality**. Suppose instead that  $L$  is not orthogonal to  $(E + e) \setminus L$ . We will show how to modify  $Q$  and  $\mathcal{M}$  and possibly the partition  $\mathcal{P} + \{e\}$  so that Invariant 1 becomes satisfied.

#### Repair-Orthogonality

INPUT:  $(E + e, S, \mathcal{P} + \{e\}, Q, \mathcal{M}, \mathcal{H})$ .

```

begin
  select a pseudomatching  $M^* \in \mathcal{M}$  so that  $e \notin \overline{M^*}$ ;
  (this exists by Invariant 3 and the fact that  $e \notin \overline{L \cup Q}$ );
  if  $M^* + e$  is a pseudomatching, then go to Procedure Transform-
    Pseudomatching;
  else, continue;
  replace  $e$  by an equivalent line  $e' = (e'_1, e'_2)$ , where  $e'_2 \in \overline{M^*}$ ;
  (linearity is assumed in this step;)
  for each line  $e_j \in L$  such that  $M^* + e' - e_j$  is a pseudomatching do
    begin
      replace  $e_j$  by an equivalent line
         $e'_j = (e'_{j1}, e'_{j2})$ , where  $e'_{j2} \in \overline{M^* + e' - e_j}$ ;
      add  $e'_{j2}$  to  $Q$  and also to  $S$ ; (thus we view  $e'_{j2}$  as a
        pseudopoint, even though it is also a point);
      add  $M^* + e' - e_j$  to  $\mathcal{M}$ ;
    end
  end
end

```

**Theorem 3.**  $(E, S, \mathcal{P}, Q, \mathcal{M}, \mathcal{H})$   $fi$   $1 \quad 4,$   
 $(E + e, S, \mathcal{P}', Q', \mathcal{M}', \mathcal{H})$   $6-$

**Repair-Orthogonality.**

$$\begin{array}{rcccl}
 & (E+e, \mathcal{P}', Q', \mathcal{M}', \mathcal{H}) & fi & 1, 3 & 4. \\
 r(e+Q') = r(Q') + 1, & 2 & fi & \cdot & r(e+Q') = r(Q') + 2, \\
 \mu(\mathcal{P}', Q') = \mu(\mathcal{P}, Q) + 1, & & 2 & & fi
 \end{array}$$

. In appendix. □

When a line  $e$  is transformed in the above algorithm, it is always replaced by an equivalent line with exactly one point in  $Q$ .

## 8 The $(\mathcal{P}, Q)$ -Matroid Intersection Problem

In this section, we describe Procedure **Matroid-Intesersection**. The input for this procedure is the output  $(E, S, \mathcal{P}, Q, \mathcal{M}, \mathcal{H})$  from Procedure **Repair-Orthogonality**, assuming that the procedure was called. We assume that  $(E, S, \mathcal{P}, Q, \mathcal{M}, \mathcal{H})$  does not satisfy Invariant 2. For each pseudomatching  $M \in \mathcal{M}$ ,  $|M| = \mu(\mathcal{P}, Q) - 2$ . We assume that  $\mathcal{P} = (E_1, \dots, E_K)$ , and we again let  $L = L(\mathcal{M})$ .

We next create a linear matroid intersection problem whose elements are a combination of “simple elements” and “compound elements.” We refer to this problem as the  $(\mathcal{P}, Q)$ - , and often refer to it more briefly as the . The output from the intersection problem induces a  $\mathcal{P}$ -matching  $M$  with  $|M| = \mu(\mathcal{P}, Q)$  or else it induces a modified dual solution  $(\mathcal{P}', Q')$  such that  $\mu(\mathcal{P}', Q') = \mu(\mathcal{P}, Q) - 2$ . In the latter case,  $(\mathcal{P}', Q')$  is a minimum capacity dual solution for  $E$ .

### The $(\mathcal{P}, Q)$ -Matroid Intersection Problem

The intersection problem is defined on a set  $W = W_S \cup W_C$ , where  $W_S$  is a set of “simple elements”, and  $W_C$  is a set of “compound elements”. For each point  $p \in E$ , there is a  $p \in W_S$ . For each odd component  $E_i$ , there is an associated point  $b_i$ , which is the first non-critical point of  $E_i$ ; i.e.,  $b_i \notin \overline{Q}$ . Let  $B = \{b_i : E_i \text{ is an odd component}\}$ . For each pseudopoint  $p$  of  $S_i$ , there is a  $b_{i-p} \in W_C$ . The compound element will be treated as if it were  $b_i$  with respect to the first matroid (defined below), and it will be treated as if it were  $p$  with respect to the second matroid (defined below).

For each set  $J \subseteq W$ , we define three subsets of points of  $E \cup S$  induced by the set  $J$ :

1.  $J_S = \{p \in E : p \text{ is a simple element of } J\}$ ;
2.  $J_B = \{b_i \in E : b_i - p \text{ is a compound element of } J \text{ for some pseudopoint } p \in S\}$ ;
3.  $J_Q = \{p \in S : b_i - p \text{ is a compound element of } J\}$ .

We next define the two matroids  $(W, \mathcal{I}_1)$  and  $(W, \mathcal{I}_2)$  for the matroid intersection problem, with collections  $\mathcal{I}_1$  and  $\mathcal{I}_2$  of independent sets, and with rank functions  $r_1()$ , and  $r_2()$ . Suppose  $J \subseteq W$ .

1.  $J \in \mathcal{S}_1$  if  $J_S \cup J_B$  is independent with respect to  $r_Q(\cdot)$ ; equivalently,  $r_1(J) = r_Q(J_S \cup J_B)$ .
2.  $J \in \mathcal{S}_2$  if  $r(J_Q) = |J_Q|$  and if  $|J_S \cap E_i| \leq 2 \lfloor r_Q(E_i)/2 \rfloor$  for all components  $E_i$ ; equivalently,

$$r_2(J) = r(J_Q) + \sum_{i=1}^K \min \{ |J_S \cap E_i|, 2 \lfloor r_Q(E_i)/2 \rfloor \}.$$

We note that  $r_2(W) = \mu(\mathcal{P}, Q) - r(Q)$ .

The intuition behind this matroid intersection problem is based on the following theorem.

**Theorem 4.**

1.  $r_2(W) = r_2(W) - 1$ .
2.  $J \subseteq W, J_S \cup J_B \cup J_Q$   
 $E \cup S, r(\cdot)$ .
3.  $J_Q, M = J_S \cup J_B \cup E, |M| = \mu(\mathcal{P}, Q)$   
 $J, |J| = r_2(W),$

. In appendix. □

If the matroid intersection algorithm does not lead to a larger cardinality intersection, then by Theorem 4, the largest intersection is of size  $r_2(W) - 1$ . By matroid intersection duality (see, for example, Lawler [5]), the algorithm provides a partition of the elements  $W$  into two subsets  $W_1$  and  $W_2$  such that  $r_1(W_1) + r_2(W_2) = r_2(W) - 1$ .

**Lemma 2.**

$$(W_1, W_2) \subseteq W, r_1(W_1) + r_2(W_2) = r_2(W) - 1.$$

$$E_j \subseteq W_1, E_j \subseteq \mathcal{P}, E_j \subseteq W_1.$$

. In appendix. □

**Theorem 5.**

$$\mathcal{P} \subseteq W_2, \mu(\mathcal{P}', Q') = \mu(\mathcal{P}, Q) - 2, r_2(W) - 1.$$

$$E \cup S, Q' \subseteq (\mathcal{P}', Q')$$

. In appendix. □

In the case that largest intersection is of size  $r_2(W) - 1$ , the algorithm produces additional matchings of size  $\mu(\mathcal{P}, Q) - 2$  so that Invariant 2 is satisfied when  $Q$  is replaced by  $Q'$ .

## 9 Transforming the Pseudomatchings

In this section, we explain how to transform a maximum cardinality pseudomatching of  $E \cup S$  into a maximum cardinality matching of  $E$ . This is the case in which the matroid intersection algorithm outputs an intersection of size  $r_2(W)$ , which in turn induces a pseudomatching  $M$  of size  $\mu(\mathcal{P}, Q)$ , which is larger than any pseudomatching of  $\mathcal{M}$ . Since each matching of  $\mathcal{M}$  is maximum in  $E \setminus e$ , it follows that  $e \in M$ , and that  $M \setminus e$  is a maximum matching in  $E \setminus e$ .

Let  $L = L(\mathcal{M})$ . By Corollary 1,  $L + e \subseteq M$ . For every odd component  $E_j$  of  $\mathcal{P}$ , let  $M_j = M \cap (E_j \cup S_j)$ . By Corollary 1,  $M_j$  is either a  $(\mathcal{P}, Q)$ -hypomatching or a  $(\mathcal{P}, Q)$ -hypermatching.

We will next show how to transform  $M$  into a matching by expanding one component at a time in a LIFO order. Consider component  $E_j$  and suppose that it was created by merging components  $E_i$  for  $i \in I$ .

**Transforming Hypomatchings.** Suppose that  $M_j$  is a hypomatching. First choose point  $p \in E_j$  such that  $r_Q(M_j + p) = |M_j| + 1$ . Actually, we choose  $p \in \overline{M'} \setminus \overline{E_j} \cap \overline{E_j} \setminus \overline{Q}$ . Now choose a pseudomatching  $M' \in \mathcal{M}$  such that  $p \notin \overline{M'}$ . (It exists because Invariant 3 is satisfied). Now replace  $M_j$  in  $M$  by  $M' \cap E_j$ , obtaining a pseudomatching  $M''$ .  $M'$  was determined by the algorithm prior to component  $E_j$  being formed, except in the case that  $E_j$  is a single line. So  $M''$  is also a pseudomatching if we expand component  $E_j$ . ( ).

**Transforming Hypermatchings.** Suppose that  $M_j$  is a hypermatching that contains  $q \in S_j$ , and that  $E_j$  is not a singleton line. Then replace  $M_j$  by the hypermatching  $H(q)$ , which was created at the time that  $E_j$  was merged into a component.

## 10 Creating Pseudopoints and Hypermatchings

If the matroid intersection algorithm does not lead to a larger pseudomatching, then it leads to merger of several components, creating a new component  $E_j$ . At the same time, the set  $Q$  is replaced by a possibly smaller set  $Q'$ . If  $\overline{E_j} \cap \overline{Q'} \neq \emptyset$ , then the algorithm runs a modified matroid intersection algorithm in order to obtain pseudopoints  $q_1, \dots, q_k \in Q'$  that forms a basis of  $\overline{E_j} \cap \overline{Q'}$ , and hypermatchings  $H(q_1), \dots, H(q_k)$  such that  $q_i \in \overline{H(q_i)}$  for  $i = 1$  to  $k$ . This step also requires the linearity of the representation. ( ).

## 11 Analysis of Running Time

We claim that the LMPP algorithm runs in  $\mathcal{O}(m^3n)$  time. This is the same running time for running the matroid intersection algorithm for linearly representable matrices. We now give a brief overview of the run time analysis. Since there are  $\mathcal{O}(m)$  augmentation phases, it suffices to show that each augmentation phase runs in  $\mathcal{O}(m^2n)$  time.



Whenever a new pseudomatching  $M$  is added to  $M$ , we extend  $M$  to a basis  $B(M)$ , and then pivot (using Gaussian elimination) so that the basis is transformed to the identity matrix. Normally, one would expect this to take  $\mathcal{O}(m^2n)$  time for each pseudomatching. However, the pseudomatchings in  $M$  are close to one another in that they differ by a very small number of points. Suppose that the pseudomatchings are  $M_1, \dots, M_k$ . Let  $p_j$  be the number of pivots needed to transform  $B(M_j)$  to the identity matrix, assuming that one can start with one of the  $j - 1$  previously transformed matrices. One can show that  $p_1 + \dots + p_k = \mathcal{O}(m)$ . That is, it requires a linear number of pivots to transform the  $\mathcal{O}(m)$  different bases into canonical form.

The run time analysis for the pivoting shows that the time for Repair-Orthogonality and Matroid Intersection both take  $\mathcal{O}(m^2n)$  time per pivoting phase. We now consider some of the other steps. The time to determine eligible lines can be reduced to  $\mathcal{O}(mn)$  time per eligible line. It requires scanning the elements of a matrix at each step. The time for creating pseudopoints and pseudomatchings is  $\mathcal{O}(m^2)$  time for every new component. It turns out that one can then directly find pseudopoints and hypermatchings efficiently once the matroid intersection problem is set up.

Finally, there is the time for transforming pseudomatchings. It turns out that the major technical difficulty arises in transforming hypomatchings, but it can be done in  $\mathcal{O}(m^2n)$  time. This is discussed in the appendix.

## 12 Summary and Conclusions

We have presented an  $\mathcal{O}(m^3n)$  algorithm for the linear matroid parity problem. Although the algorithm is complex, it is far simpler than the algorithm of Orlin and Vande Vate [10], on which this algorithm is based. We also believe that it is simpler than other algorithms for the LMPP. Moreover, the algorithm presented here achieves the same running time as that of Gabow and Stallman [3], which is the fastest time for the LMPP.

Perhaps surprisingly, this LMPP algorithm also provides a new algorithm for the matroid intersection problem, even if the two matroids are not representable. If one runs the LMPP algorithm starting with a matroid intersection problem, then there is no need for representability. One place in which the LMPP algorithm relies on linear representability is when components are merged, but this merger never occurs in a matroid intersection problem. The other place in LMPP where linearity is required is during **Repair-Orthogonality**. But linearity is not required when the algorithm is specialized to matroid intersection, and there is no need to “transform lines.” When LMPP is applied to the matroid intersection problem, it results in a different algorithm than those developed by Lawler [5] because it is incremental, adding one element at a time, rather than creating a single auxiliary graph for each augmentation phase. As such, it has a slightly different run time analysis. For linear matroids, it has the same running time as the previous algorithms.

## References

1. Edmonds, J.: Paths, trees, and flowers. *Canadian Journal of Mathematics* 17, 449–467 (1965)
2. Edmonds, J.: Submodular functions, matroids, and certain polyhedral. In: Guy, R., Hanani, H., Sauer, N., Schönheim, J. (eds.) *Combinatorial Structures and Their Applications* (proceedings Calgary International Conference on Combinatorial Structures and Their Applications, Calgary, Alberta, 1969), pp. 69–87. Gordon and Breach, New York (1970)
3. Gabow, H.N., Stallmann, M.: An augmenting path algorithm for linear matroid parity. *Combinatorica* 6, 123–150 (1986)
4. Jensen, P.M., Korte, B.: Complexity of matroid property algorithms. *SIAM Journal on Computing* 11, 184–190 (1982)
5. Lawler, E.L.: Optimal matroid intersections. In: Guy, R., Hanani, H., Sauer, N., Schönheim, J. (eds.) *Combinatorial Structures and Their Applications* (proceedings Calgary International Conference on Combinatorial Structures and Their Applications, Calgary, Alberta, 1969), pp. 233–234. Gordon and Breach, New York (1970)
6. Lawler, E.L.: *Combinatorial Optimization: Networks and Matroids*. Hold, Rinehart and Winston, New York (1976)
7. Lovász, L.: Matroid matching and some applications. *Journal of Combinatorial Theory Series B* 28, 208–236 (1980)
8. Lovász, L.: The matroid matching problem. In: Lovász, L., Sós, V.T. (eds.) *Algebraic Methods in Graph Theory, Vol. II* (Colloquium Szeged, 1978). *Colloquia, Mathematica Societatis János Bolyai*, vol. 25, pp. 495–517. North-Holland, Amsterdam (1981)
9. Lovász, L., Plummer, M.D.: *Matching Theory*. Akadémiai Kiadó, Budapest (1986) (also North-Holland Mathematics Studies vol. 121. North-Holland, Amsterdam)
10. Orlin, J.B., Vande Vate, J.H.: Solving the matroid parity problem as a sequence of matroid intersection problems. *Mathematical Programming* 47, 81–106 (1990)
11. Schrijver, A.: *Combinatorial Optimization: Polyhedra and Efficiency*, vol. 2. Springer, New York (2003)
12. Stallmann, M., Gabow, H.N.: An augmenting path algorithm for the parity problem on linear matroids. In: *25th Annual Symposium on Foundations of Computer Science*, New York, pp. 217–227. IEEE, Los Alamitos (1984)

## Appendix

**Theorem 3.**  $(E, S, \mathcal{P}, Q, \mathcal{M}, \mathcal{H})$   $\hat{f}_i$   $1$   $4$ ,  
 $(E + e, S, \mathcal{P}', Q', \mathcal{M}', \mathcal{H})$   $6$ -

**Repair-Orthogonality.**

$$r(e + Q') = r(Q') + 1, \quad (E + e, \mathcal{P}', Q', \mathcal{M}', \mathcal{H}) \quad \hat{f}_i \quad 1, 3 \quad 4.$$

$$\mu(\mathcal{P}', Q') = \mu(\mathcal{P}, Q) + 1, \quad \hat{f}_i \quad 2 \quad \hat{f}_i \quad r(e + Q') = r(Q') + 2, \quad \hat{f}_i \quad 2 \quad \hat{f}_i \quad .$$

. Let  $L' = L(M')$ . We first consider property Invariant 4, which concerns hypermatchings. Let  $E_j$  be a non-trivial odd component of  $\mathcal{P}'$ . Since no non-trivial components were formed during **Repair-Orthogonality**,  $E_j$  is also a

non-trivial component of  $\mathcal{P}$ . By Invariant 1,  $L$  is orthogonal to  $E \setminus L$ , and thus is orthogonal to  $E_j$ . Therefore,  $\overline{E_j} \cap \overline{Q'} = \overline{E_j} \cap \overline{Q}$ , and so Invariant 4 remains satisfied.

We next consider Invariant 3, which concerns the intersection of matchings in  $\mathcal{M}'$ . For convenience, let us reorder the elements of  $E$  (and thus  $E^*$ ) so that the matchings added to  $\mathcal{M}$  are  $M_i^* = (M^* + e) \setminus e_i$  for  $i = 1$  to  $t$ . For convenience, we are also assuming that the lines do not need to be transformed. Then  $\mathcal{M}' = \mathcal{M} \cup \{M_1^*, \dots, M_t^*\}$ ,  $Q' = Q \cup \{e_{12}, \dots, e_{t2}\}$ , and  $L' = L \setminus \{e_1, \dots, e_t\}$ . To see that  $Q' \subseteq \bigcap_{M \in \mathcal{M}'} \overline{M \setminus L'}$ , note first that  $Q' \subseteq \overline{L \cup Q}$  because  $Q'$  is obtained by taking points from  $L$  and adding them to  $Q$ . Therefore, for all  $M \in \mathcal{M}$ ,  $Q' \subseteq \overline{M}$ . The remaining matchings of  $\mathcal{M}'$  are of the form  $(M^* + e) \setminus e_j$ . In this case,  $Q \subseteq \overline{(M^* + e) \setminus e_j}$  because  $e_j$  was a line of  $L$ , which is orthogonal to  $Q$ ; and  $Q' \setminus Q \subseteq \overline{(M^* + e) \setminus e_j}$ , because every point added to  $Q'$  was a point of  $\overline{M^* + e - e_j}$  for some  $j = 1$  to  $t$ . Thus,  $Q' \subseteq \bigcap_{M \in \mathcal{M}'} \overline{M \setminus L'}$ .

To complete the proof that Invariant 3 is satisfied by  $(E', S, \mathcal{P}', Q', \mathcal{M}', \mathcal{H})$ , we need to show that if  $p \notin \overline{L' \cup Q'}$ , then there is some matching  $M \in \mathcal{M}'$  so that  $p \notin \overline{M \setminus L'}$ . If  $p \notin \overline{L \cup Q}$ , then by assumption, there is a matching  $M \in \mathcal{M}$  so that  $p \notin \overline{M \setminus L} \supseteq \overline{M \setminus L'}$ . The remaining case is  $p \in \overline{L \cup Q}$  and  $p \notin \overline{L' \cup Q'}$ . Without loss of generality, we may assume that  $p \in \overline{L}$ . (Otherwise, since  $L$  and  $Q$  are orthogonal, we can express  $p$  as  $p_1 + p_2$ , where  $p_1 \in \overline{L}$  and  $p_2 \in \overline{Q}$ . Since  $p_2 \in \overline{Q} \subseteq \overline{Q'}$  and  $p_1 + p_2 \notin \overline{Q'}$ , it follows that  $p_1 \notin \overline{Q'}$ . And if  $p \notin \overline{M \setminus L'}$  then  $p_1 \notin \overline{M \setminus L'}$ .) Adding  $p$  to  $L$  creates a circuit containing some point  $e_{j1} \notin Q'$  for some  $j = 1$  to  $t$  because  $p \notin \overline{L'}$ . Then  $p \notin \overline{M_j}$ , completing the proof that Invariant 3 remains satisfied.

We next consider Invariant 1. Since  $L'$  is obtained from  $L$  by deleting elements, clearly  $L'$  is a matching. We next prove orthogonality of  $L'$  to  $(E + e) \setminus L'$ , assuming that  $M^* + e$  is not a larger matching than  $M^*$ .

Let  $B$  be of minimal set of points of  $(E + e) \setminus L$  so that  $\overline{B \cup M^* \cup e} = \overline{E + e}$ . Then (i)  $B$  is orthogonal to  $M^*$ , (ii)  $r(B \cup M^*) = r(E') - 1$ , and (iii)  $r(B \cup M^* \cup \{e\}) = r(E')$ . Thus the unique circuit in  $B \cup M^* \cup \{e\}$  is  $C$ , and  $C \subseteq (B \cup \{e\} \cup M^*) \setminus L'$ . Thus,  $r(L') + r(B \cup \{e\} \cup M^* \setminus L') = r(L') + r((E + e) \setminus L') = r(E + e)$ , and  $L'$  is orthogonal to  $(E + e) \setminus L'$ , completing the proof that Invariant 1 is satisfied.

Finally we consider Invariant 2. Every point  $e_{j2}$  added to  $Q'$  has no effect on the capacity of the dual solution because it subtracts 1 from  $r_Q(e_j)/2$ , and adds 1 to  $r(Q)$ . The possible increase in the capacity of the dual solution is due to the singleton component containing  $e$ . If  $r_{Q'}(e) = 2$ , then  $\mu(\mathcal{P}', Q') = \mu(\mathcal{P}, Q) + 2$  and Invariant 2 is not satisfied. If  $r_{Q'}(e) = 1$ , then  $\mu(\mathcal{P}', Q') = \mu(\mathcal{P}, Q)$ , and Invariant 2 is satisfied. □

**Theorem 4.**

$$\begin{array}{llll}
 1. & & r_2(W) & r_2(W) - 1. \\
 2. & J & W, & J_S \cup J_B \cup J_Q \\
 & & E \cup S & r().
 \end{array}$$

$$\begin{array}{llll}
 3. & & J & |J| = r_2(W), & M = J_S \cup J_B \cup \\
 & J_Q. & M & & E, & |M| = \\
 & \mu(\mathcal{P}, Q). & & & & 
 \end{array}$$

. We first prove (1). Let  $M^* \in \mathcal{M}$  be chosen so that  $r(M^* + e) = r(M^*) + 1$ . Such a matching exists because  $e$  is eligible. By Invariant 2,  $|M^*| = \mu(\mathcal{P}, Q)$ . Let  $q$  be a point of  $e$  so that  $M^* + q$  is independent. We next transform  $M^* + q$  into a feasible intersection as follows:

begin

```

M' := M* + q;
I := {q};
for each line e_j of L, I := I + e_{j1} + e_{j2};
for each odd component such that E_j \cap M* is a hypomatching,
  I := I \cup (E_j \cap M*);
for each odd component E_i for which M_i = E_j \cap M* is a
  hypermatching, replace one of the points of M_i by the point p
  in \overline{M_i} \cap \overline{Q}, and replace one of the points in M_i by b_i so as to
  maintain independence; then add b_i - p to I and add the
  remaining points of M_i;

```

end

Then  $|I| = r(M^*) + 1 - r(Q)$ . The term “ $r(Q)$ ” is due to the fact that each point in  $Q$  is transformed into a compound point in  $I$ . It follows that  $|I| = r_2(W) - 1$ . (The “ $-1$ ” term comes from the fact that a basis with respect to  $r_2(\cdot)$  contains both points of  $e$ , and  $I$  contains just one point of  $e$ ). Thus (1) is true.

We next show that  $J_S \cup J_B \cup J_Q$  is an independent set of points. If  $J$  is any feasible intersection of  $W$ , then  $r_1(J) = r_Q(J_S \cup J_B) = |J_S \cup J_B|$ . Because  $r_2(J) = |J|$ , it follows that  $r(J_Q) = |J_Q|$ . Therefore,  $J_S \cup J_B \cup J_Q$  is an independent set of points of  $E \cup \overline{Q}$  with respect to  $r(\cdot)$ .

We now show that if  $|J| = r_2(W)$ , then  $M = J_S \cup J_B \cup J_Q$  is a maximum cardinality  $\mathcal{P}$ -matching of  $E$ . By (2),  $M$  is independent. Also,  $M = J_S \cup J_B \cup J_Q = r_2(W) + r(Q) = \mu(\mathcal{P}, Q)$ . It follows that  $M$  is a maximum cardinality matching. □

**Lemma 2.**

$$\begin{array}{llll}
 (W_1, W_2) & & W & r_2(W) - 1. \\
 r_1(W_1) + r_2(W_2) = r_2(W) - 1. & & E_j & \mathcal{P}, \\
 & & E_j & W_1.
 \end{array}$$

. Suppose conversely that point  $p_1 \in E_j \cap W_1$  and  $p_2 \in E_j \cap W_2$ . In this case, transferring  $p_2$  from  $W_2$  to  $W_1$  cannot increase the sum  $r_1(W_1) + r_2(W_2)$  because the transfer must decrease  $r_2(W_2)$ . In this way, we can transfer all the remaining simple elements of  $E_j \cap W_2$  from  $W_2$  to  $W_1$ , while maintaining the matroid intersection optimality conditions. □

**Theorem 5.**

$$\begin{array}{llll}
 \mathcal{P}' & & \mathcal{P} & r_2(W) - 1. \\
 & & W_1. & Q'
 \end{array}$$

$$W_2. \quad \mu(\mathcal{P}', Q') = \mu(\mathcal{P}, Q) - 2, \quad (\mathcal{P}', Q') \\ E \cup S.$$

. Let  $J$  be a maximum cardinality intersection. By matroid intersection duality,  $|J| = r_1(W_1) + r_2(W_2) = r_2(W) - 1 = \mu(\mathcal{P}, Q) - r(Q) - 1$ . (An intersection of size  $r_2(W)$  would have led to an augmentation).

We next derive a formula for  $r_{Q'}(\bigcup_{E_i \in W_1} E_i)$ . First note that

$$|J| = r_1(W_1) + r_2(W_2) = r_Q \left( \bigcup_{E_i \in W_1} E_i \right) + r(Q') + 2 \sum_{E_i \in W_2} \lfloor r_Q(E_i)/2 \rfloor. \quad (1)$$

Also,

$$|J| = r_2(W) - 1 = r(Q) - 1 + 2 \sum_{i=1}^K \lfloor r_Q(E_i)/2 \rfloor. \quad (2)$$

Combining (1) and (2) yields:

$$r_Q \left( \bigcup_{E_i \in W_1} E_i \right) = r(Q) - r(Q') - 1 + 2 \sum_{E_i \in W_1} \lfloor r_Q(E_i)/2 \rfloor. \quad (3)$$

Moreover,  $r_Q(E_j) = r_{Q'}(E_j)$  for  $E_j \in W_2$ , and for any set  $A \subseteq \mathbb{R}_m$ ,  $r_{Q'}(A) = r_Q(A) + r(Q) - r(Q')$ . Substituting into (3) we get

$$r_{Q'} \left( \bigcup_{E_i \in W_1} E_i \right) \leq 2r(Q) - 2r(Q') - 1 + 2 \sum_{E_i \in W_1} \lfloor r_Q(E_i)/2 \rfloor,$$

and

$$\left\lfloor r_{Q'} \left( \bigcup_{E_i \in W_1} E_i \right) / 2 \right\rfloor = \left( r(Q) - r(Q') + \sum_{E_i \in W_1} \lfloor r_Q(E_i)/2 \rfloor \right) - 1.$$

Therefore,

$$\mu(\mathcal{P}', Q') = 2r(Q') + 2 \left\lfloor r_{Q'} \left( \bigcup_{E_i \in W_1} E_i \right) / 2 \right\rfloor + 2 \sum_{E_i \in W_2} \lfloor r_{Q'}(E_i)/2 \rfloor \\ \leq \left( 2r(Q) + 2 \sum_{E_i \in W} \lfloor r_Q(E_i)/2 \rfloor \right) - 2 = \mu(\mathcal{P}, Q) - 2. \quad \square$$

### Solving the Matroid Intersection Problem

Here we describe the auxiliary graph that is used in the matroid intersection algorithm. The results are used in the next section to describe how to obtain additional pseudomatchings for  $\mathcal{M}$  as well as creating the hypermatchings for  $\mathcal{H}$ .

We solve the matroid intersection problem using the augmenting path theorem of Edmonds [2] and Lawler [5]. Let  $\oplus$  denote the  $\text{ff}$  . That is,  $X \oplus Y = (X \setminus Y) \cup (Y \setminus X)$ .

Let  $J$  be an initial feasible intersection. We say that  $J \oplus I$  is obtained via an  $\mathcal{S}_1$ - $\mathcal{S}_2$  if the following is true:

- for each odd  $q$ ,  $J + i_1 - i_2 + \dots - i_{q-1} + i_q \in \mathcal{S}_1$ ,
- for each even  $q$ ,  $J + i_1 - i_2 + \dots + i_{q-1} - i_q \in \mathcal{S}_2$ .

Suppose that  $J \in \mathcal{S}_1 \cap \mathcal{S}_2$ , and let  $I = i_1, i_2, i_3, \dots, i_k$  be elements of  $W$ , where  $i_q \in E \setminus J$  for  $q$  odd, and  $i_q \in J$  for  $q$  even. If in addition,  $k$  is odd and  $J + i_1 - i_2 + \dots - i_{k-1} + i_k \in \mathcal{S}_2$ , then we refer to the alternating path as an  $\mathcal{S}_1$ - $\mathcal{S}_2$  . Usually, we refer to them more briefly as

and . If  $J \oplus I$  is an augmenting path, then the subset  $J \oplus I$  is a feasible intersection that is larger than  $J$ . If  $J$  is a feasible intersection that is not of maximum cardinality, then there is an  $\mathcal{S}_1$ - $\mathcal{S}_2$  augmenting path with respect to  $J$ .

The Matroid Intersection Algorithm finds an  $\mathcal{S}_1$ - $\mathcal{S}_2$  by finding a path in a graph called the  $G = (E, A)$  defined as follows:

1. The graph is bipartite; the two nodes sets are  $E \setminus J$  and  $J$ .
2. The node  $i \in E \setminus J$  is a destination node if  $r_2(J + i) = |J| + 1$ . Node  $i \in E \setminus J$  is a source node if  $r_1(J + i) = |J| + 1$ .
3. If  $i \in E \setminus J$  is not a destination node, then for  $j \in J$ , there is an arc  $(i, j) \in A$  if  $r_1(J + i - j) = |J|$ .
4. If  $i \in E \setminus J$  is not a source node, then for  $j \in J$ , there is an arc  $(j, i) \in A$  if  $r_2(J + i - j) = |J|$ .

The following theorem is well known; see for example, Lawler [6].

**Theorem 6 (Auxiliary Graph Theorem).** Let  $P = i_1, i_2, \dots, i_k$  be an  $\mathcal{S}_1$ - $\mathcal{S}_2$  augmenting path with respect to  $J$ . Let  $G = (E, A)$  be the auxiliary graph defined above. Then  $|J \oplus P| = |J| + 1$ . Let  $I = \{i \in E : (i, j) \in A \text{ for some } j \in J \text{ and } r_1(J + i - j) = |J|\}$ . Then  $|J| = r_1(I) + r_2(E \setminus I)$ . □

The following theorem is also an immediate consequence of the standard proof given for Theorem 6.

**Theorem 7 (Alternative Maximum Matchings).** Let  $G = (E, A)$  be the auxiliary graph defined above. Let  $I = \{i \in E : (i, j) \in A \text{ for some } j \in J \text{ and } r_1(J + i - j) = |J|\}$ . Let  $P(k) = \{i \in E : (i, j) \in A \text{ for some } j \in J \text{ and } r_2(J + i - j) = |J| + 1\}$ . Then  $J(k) = J \oplus P(k)$  is a maximum cardinality intersection. □

**Creating New Hypermatchings and Pseudomatchings**

Let  $J$  be the intersection at the beginning of the matroid intersection algorithm, and let  $b_i - q \in J \cap W_1$ . Let  $P$  be the shortest alternating path in  $G$  from a source node to  $b_i - q$ . Then  $J \oplus P$  is also a max cardinality intersection, and

it induces a matching  $M'$  that is added to  $\mathcal{M}$ . Previously,  $q$  was in the span of every matching in  $\mathcal{M}$ . But  $q \notin \overline{M'}$ . Similarly, for all pseudopoints  $q' \in S \cap W_1$ , there is a matching  $M''$  added to  $\mathcal{M}$  so that  $q' \notin \overline{M''}$ .

We now consider the component  $E_{K+1}$  induced by  $W_1$ . We next point out how the algorithm determines hypermatchings for the newly formed component. Let  $Q'' = S \cap W_1$ . Now consider the matroid intersection problem obtained by restricting attention to the components in  $W_1$ , and then looking for a maximum cardinality intersection defined with rank functions  $r'_1$  and  $r'_2$ , where

1.  $r'_1(J) = r_{Q''}(J_S \cup J_B)$ .
2.  $r'_2(J) = r(J_{Q''}) + \sum_{E_i \in W_1} \min \{|J_S \cap E_i|, 2 \lfloor r_Q(E_i)/2 \rfloor\}$ .

The maximum intersections all induce hypermatchings of  $E_{K+1}$ . For each hypermatching  $M'$  formed in this manner, we select  $q \in \overline{M'} \cap \overline{Q'}$ , we add the pseudopoint  $q$  to  $S_{K+1}$ , and we let  $H(q) = M'$ .

### Analysis of the Running Time

We discuss two aspects of the running time. The first aspect concerns the time it takes to carry out all of the pivoting so that each pseudomatching in  $\mathcal{M}$  is in canonical form. The second concerns the time it takes to treat hypomatchings when transforming a pseudomatching into a matching.

Consider when pseudomatchings are added to  $\mathcal{M}$ . We need to bring all of these pseudomatchings into canonical form. We claim we can do so with  $\mathcal{O}(m)$  pivots.

In **Repair-Orthogonality**, we create pseudomatchings of the form  $M^* + e - e_j$ , where  $M^* \in \mathcal{M}$ . Canonical form for each of these pseudomatchings can be created from canonical form from  $M^*$  with two additional pivots. Thus, each pseudomatching of this type leads to  $\mathcal{O}(1)$  additional pivots.

We next consider pseudomatchings added to  $\mathcal{M}$  during **Matroid-Intersection**. Let  $T$  be the shortest path tree from the unique source node of the auxiliary graph, and let  $|T|$  be the number of nodes of  $T$ . Then  $|T|$  is at most twice the number of components that were merged to create  $E_{K+1}$ . The number of pivots needed to bring all of the newly created matchings is at most  $2|T|$ . Since the total merging of components over all iterations is  $\mathcal{O}(m)$ , it follows that the number of pivots needed to bring all matchings of  $M$  into canonical form is  $\mathcal{O}(m)$ , which is what we wanted to show.

We now consider the time it takes to identify hypomatchings within **Transform Pseudomatching**. Suppose that  $M'$  a current pseudomatching, and we want to expand component  $E_j$ . We expand components in a LIFO ordering. That is, the last component formed is the first one expanded.

Suppose that  $|M' \cap E_j| = r_Q(E_j) - 1$ . Possibly  $M'$  is not a feasible pseudomatching after we expand  $E_j$ . We now consider two subcases. In the first subcase, there is a point  $p \in E_j$  such that  $r(M' + p) = r(M') + 1$ . Then  $M' \cap E_j$  can be replaced by any hypomatching of  $E_j$ . In particular, it can be replaced by a hypomatching in  $M \cap E_j$ , where  $M$  is a matching in  $\mathcal{M}$ . Moreover, in this

case, we can expand the component  $E_j$  into a collection of components, each consisting of a single line.

In the second subcase,  $E_j \subseteq \overline{M'}$ . In this case, we select  $p \in \overline{M' \setminus E_j} \cap \overline{E_j \setminus Q}$ . Such a point exists and can be found in  $\mathcal{O}(m^2)$  time. We then let  $M$  be a matching in  $\mathcal{M}$  that does not contain  $p$ . We then replace  $M'$  by  $M'' = (M' \setminus E_j) \cup (M \cap E_j)$ . This can be accomplished in  $\mathcal{O}(m^2)$  time. We now claim that  $M''$  is a pseudomatching. We note that  $p \in \overline{M' \setminus E_j} \subseteq \overline{M''}$ , and so

$$\begin{aligned} r(M'') &= r(M'' + p) = r((M' \setminus E_j) \cup ((M + p) \cap E_j)) \\ &= r((M' \setminus E_j) \cup E_j) = r(M'). \end{aligned}$$

Moreover, if we let  $M$  be the first pseudomatching in  $\mathcal{M}$  not containing  $p$ , then  $M$  was created prior to merging  $E_j$ , and so  $M''$  is a pseudomatching after expanding  $E_j$ . And the time it takes to find the first pseudomatching in  $\mathcal{M}$  not containing  $p$  is  $\mathcal{O}(m^2)$ .

We conclude that each hypomatching can be expanded in  $\mathcal{O}(m^2)$  time, with a total time of  $\mathcal{O}(m^3)$  per augmentation phase, which is what we wanted to show.



# Degree Bounded Matroids and Submodular Flows

Tamás Király<sup>1,\*</sup>, Lap Chi Lau<sup>2</sup>, and Mohit Singh<sup>3,\*\*</sup>

<sup>1</sup> MTA-ELTE Egerváry Research Group, Dept. of Operations Research,  
Eötvös Loránd University, Budapest

tkiraly@cs.elte.hu

<sup>2</sup> Dept. of Computer Science and Engineering, The Chinese University of Hong Kong  
chi@cse.cuhk.edu.hk

<sup>3</sup> Tepper School of Business, Carnegie Mellon University  
mohits@andrew.cmu.edu

**Abstract.** We consider two related problems, the MINIMUM BOUNDED DEGREE MATROID BASIS problem and the MINIMUM BOUNDED DEGREE SUBMODULAR FLOW problem. The first problem is a generalization of the MINIMUM BOUNDED DEGREE SPANNING TREE problem: we are given a matroid and a hypergraph on its ground set with lower and upper bounds  $f(e) \leq g(e)$  for each hyperedge  $e$ . The task is to find a minimum cost basis which contains at least  $f(e)$  and at most  $g(e)$  elements from each hyperedge  $e$ . In the second problem we have a submodular flow problem, a lower bound  $f(v)$  and an upper bound  $g(v)$  for each node  $v$ , and the task is to find a minimum cost 0-1 submodular flow with the additional constraint that the sum of the incoming and outgoing flow at each node  $v$  is between  $f(v)$  and  $g(v)$ . Both of these problems are NP-hard (even the feasibility problems are NP-complete), but we show that they can be approximated in the following sense. Let OPT be the value of the optimal solution. For the first problem we give an algorithm that finds a basis  $B$  of cost no more than OPT such that  $f(e) - 2\Delta + 1 \leq |B \cap e| \leq g(e) + 2\Delta - 1$  for every hyperedge  $e$ , where  $\Delta$  is the maximum degree of the hypergraph. If there are only upper bounds (or only lower bounds), then the violation can be decreased to  $\Delta - 1$ . For the second problem we can find a 0-1 submodular flow of cost at most OPT where the sum of the incoming and outgoing flow at each node  $v$  is between  $f(v) - 1$  and  $g(v) + 1$ . These results can be applied to obtain approximation algorithms for different combinatorial optimization problems with degree constraints, including the MINIMUM CROSSING SPANNING TREE problem, the MINIMUM BOUNDED DEGREE SPANNING TREE UNION problem, the MINIMUM BOUNDED DEGREE DIRECTED CUT COVER problem, and the MINIMUM BOUNDED DEGREE GRAPH ORIENTATION problem.

## 1 Introduction

In this paper we consider combinatorial optimization problems with degree constraints, for which the corresponding feasibility problem is already NP-complete.

---

\* Research supported by OTKA K60802 and OMFB-01608/2006.

\*\* Research supported by NSF grant CCF-0728841.

One approach to deal with these problems is to allow a slight violation of the degree constraints, and find a solution of this relaxation that has small cost. A prime example of this approach is the MINIMUM BOUNDED DEGREE SPANNING TREE problem, where we have upper (and possibly lower) bounds on the degree of the spanning tree at each node. The corresponding feasibility problem is NP-complete since it includes the Hamiltonian path problem. Goemans [8] showed that if the value of the optimal solution is  $\text{OPT}$ , then one can find in polynomial time a spanning tree of cost at most  $\text{OPT}$  that violates the degree bounds by at most 2. Using the iterative relaxation method, which is also the main technique in the present paper, Singh and Lau [12] gave an algorithm that finds a spanning tree of cost at most  $\text{OPT}$  that violates the bounds by at most 1. The aim of this paper is to obtain similar results for more general combinatorial optimization problems.

### 1.1 Minimum Bounded Degree Matroid Basis

The first problem considered is the MINIMUM BOUNDED DEGREE MATROID BASIS problem, which is a generalization of the MINIMUM BOUNDED DEGREE SPANNING TREE problem. We are given a matroid  $M = (V, \mathcal{I})$ , a cost function  $c : V \rightarrow \mathbb{R}$ , a hypergraph  $H = (V, E)$ , and lower and upper bounds  $f(e)$  and  $g(e)$  for each hyperedge  $e \in E(H)$ . The task is to find a basis  $B$  of minimum cost such that  $f(e) \leq |B \cap e| \leq g(e)$  for each hyperedge  $e \in E(H)$ . One motivation for considering the matroid generalization was the following problem posed by Frieze [7]: “Given a binary matroid  $M_A$  over the columns of a 0, 1-matrix  $A$  and bounds  $g_i$  for each row  $i$  of  $A$ , find a basis  $B$  of matroid  $M_A$  such that there are at most  $g_i$  ones in any row among columns in  $B$ ”.

A problem similar to ours has been considered recently by Chaudhuri et al. [3]. The results we give in this paper improve considerably their approximation guarantees. Our first main result is the following:

**Theorem 1.** *There exists a polynomial time algorithm for the MINIMUM BOUNDED DEGREE MATROID BASIS problem which returns a basis  $B$  of cost at most  $\text{OPT}$  such that  $f(e) - 2\Delta + 1 \leq |B \cap e| \leq g(e) + 2\Delta - 1$  for each  $e \in E(H)$ . Here  $\Delta = \max_{v \in V} |\{e \in E(H) : v \in e\}|$  is the maximum degree of the hypergraph  $H$  and  $\text{OPT}$  is the cost of an optimal solution which satisfies all the degree constraints.*

This theorem can be improved if only upper bounds (or only lower bounds) are present. The proof of the improvement uses the proof technique of Bansal et al. [1], who worked independently on the MINIMUM CROSSING SPANNING TREE problem and obtained the following result for that special case.

**Theorem 2.** *There exists a polynomial time algorithm for the MINIMUM BOUNDED DEGREE MATROID BASIS problem with only upper bounds which returns a basis  $B$  of cost at most  $\text{OPT}$  such that  $|B \cap e| \leq g(e) + \Delta - 1$  for each  $e \in E(H)$ . An analogous result holds when only lower bounds are present.*

It should be noted that this does not match the result of Singh and Lau [12] on minimum bounded degree spanning trees, since that result violates the degree

bounds by at most 1 even when both upper and lower bounds are present. Somewhat surprisingly, we show an example at the end of Section 3 indicating that obtaining a result for general matroids which satisfies both upper and lower degree bounds within additive error of one does not follow from current techniques.

### 1.2 Minimum Bounded Degree Submodular Flow

The second problem considered in this paper is the MINIMUM BOUNDED DEGREE SUBMODULAR FLOW problem. Here we are given a digraph  $D = (V, E)$ , a crossing submodular set function  $b : 2^V \rightarrow \mathbb{Z} \cup \{+\infty\}$ , node sets  $V_f \subseteq V$  and  $V_g \subseteq V$ , and functions  $f : V_f \rightarrow \mathbb{Z}_+$  and  $g : V_g \rightarrow \mathbb{Z}_+$ . Let us introduce the following notation for the set of edges entering or leaving a node set:

$$\begin{aligned} \delta^{in}(X) &= \{uv \in E : u \notin X, v \in X\}, \\ \delta^{out}(X) &= \{uv \in E : u \in X, v \notin X\}, \\ \delta(X) &= \delta^{in}(X) \cup \delta^{out}(X). \end{aligned}$$

If  $F \subseteq E$  is an edge set and  $x : E \rightarrow \mathbb{R}$  is a function on the edges, then we use the notation  $x(F) = \sum_{e \in F} x(e)$ . A *degree-constrained 0-1 submodular flow* is a vector  $x \in E \rightarrow \{0, 1\}$  with the following properties:

$$x(\delta^{in}(X)) - x(\delta^{out}(X)) \leq b(X) \quad \text{for every } X \subseteq V, \tag{1}$$

$$x(\delta(v)) \geq f(v) \quad \text{for every } v \in V_f, \tag{2}$$

$$x(\delta(v)) \leq g(v) \quad \text{for every } v \in V_g. \tag{3}$$

If  $V_f = V_g = \emptyset$ , then this is the well-studied submodular flow problem, introduced by Edmonds and Giles [5]. There are several efficient algorithms for finding a feasible submodular flow, or even a minimum cost submodular flow for a linear cost function. However, the addition of the degree constraints (2) and (3) makes the feasibility problem NP-complete, as we show in subsection 4.1. Our second main result is the following:

**Theorem 3.** *There exists a polynomial time algorithm for the MINIMUM BOUNDED DEGREE SUBMODULAR FLOW problem which returns a 0-1 submodular flow of cost at most OPT that violates each degree constraint by at most one, where OPT is the cost of an optimal solution which satisfies all the degree constraints.*

In Section 2, we show some applications of the main results. Then we present the proofs of the main results and some corresponding hardness results in Section 3 for the matroid problem and in Section 4 for the submodular flow problem.

## 2 Applications

In this section we highlight some applications of the main results.

### 2.1 Minimum Crossing Spanning Tree

In the MINIMUM CROSSING SPANNING TREE problem, we are given a graph  $G = (V, E)$  with edge cost function  $c$ , a collection of cuts (edge subsets)

$\mathcal{C} = \{C_1, \dots, C_m\}$  and bound  $g_i$  for each cut  $C_i$ . The task is to find a tree  $T$  of minimum cost such that  $T$  contains at most  $g_i$  edges from cut  $C_i$ . See [2] for various applications of this problem. The MINIMUM BOUNDED DEGREE SPANNING TREE problem is the special case where  $\mathcal{C} = \{\delta(v) : v \in V\}$ . The following result [1] (see also [11]) can be obtained as a corollary of Theorem 2. Note that  $d = 2$  for the MINIMUM BOUNDED DEGREE SPANNING TREE problem.

**Corollary 1.** [1] *There exists a polynomial time algorithm for the MINIMUM CROSSING SPANNING TREE problem that returns a tree  $T$  with cost at most  $\text{OPT}$  and such that  $T$  contains at most  $g_i + d - 1$  edges from cut  $C_i$  for each  $i$  where  $d = \max_{e \in E} |\{C_i : e \in C_i\}|$ . Here  $\text{OPT}$  is the cost of an optimal solution which satisfies all the cut constraints.*

*Proof.* Let  $M = (E, \mathcal{I})$  denote the graphic matroid over the graph  $G$ . The hypergraph  $H$  is defined with  $V(H) = E(G)$  and  $E(H) = \{C_i : 1 \leq i \leq m\}$ . Note that  $\Delta = \max_{v \in V(H)} |\{e \in E(H) : v \in e\}| = \max_{e \in E(G)} |\{C_i : e \in C_i\}| = d$ . So, using Theorem 2, we obtain a basis  $T$  of matroid  $M$  (which is a spanning tree), such that  $|T \cap C_i| \leq g_i + d - 1$ .  $\square$

## 2.2 Minimum Bounded-Ones Binary Matroid Basis

For the MINIMUM BOUNDED-ONES BINARY MATROID BASIS problem posted by Frieze [7], we are given a binary matroid  $M_A$  over the columns of a 0, 1-matrix  $A$  and bounds  $g_i$  for each row  $i$  of  $A$ . The task is to find a minimum cost basis  $B$  of matroid  $M_A$  such that there are at most  $g_i$  ones in any row among columns in  $B$ . The following result is obtained as a corollary of Theorem 2.

**Corollary 2.** *There exists a polynomial time algorithm for the MINIMUM BOUNDED-ONES BINARY MATROID BASIS problem which returns a basis  $B$  of cost at most  $\text{OPT}$  such that there are at most  $g_i + d - 1$  ones in any row restricted to columns of  $B$ . Here  $d$  is the maximum number of ones in any column of  $A$  and  $\text{OPT}$  is the cost of an optimal solution satisfying all the row constraints.*

*Proof.* Let  $M = M_A$  and define a hypergraph  $H$  where the vertex set is the columns of  $A$ . The hyperedges correspond to rows of  $A$  where  $e_i = \{A^j : A_{ij} = 1\}$  where  $A^j$  is the  $j^{\text{th}}$  column of  $A$ . Note that  $\Delta = \max_{v \in V(H)} |\{e \in E(H) : v \in e\}| = \max_j |\{i : a_{ij} = 1\}| = d$ , which is the maximum number of ones in any column of  $A$ . So, using Theorem 2, we obtain a basis of  $M = M_A$  such that number of ones in any row is at most  $g_i + d - 1$ .  $\square$

## 2.3 Minimum Bounded Degree Spanning Tree Union

In the MINIMUM BOUNDED DEGREE SPANNING TREE UNION problem, we are given a graph  $G = (V, E)$  with edge cost function  $c$ , a positive integer  $k$ , and lower and upper degree bounds  $f(v)$  and  $g(v)$  for each vertex  $v$ . The task is to

---

<sup>1</sup> Independent of the work in [11], we obtained Corollary 1 with a weaker bound using Theorem 1.

find a subgraph  $H$  which is the union of  $k$  edge-disjoint spanning trees and the degree of  $v$  in  $H$  is between  $f(v)$  and  $g(v)$ . The MINIMUM BOUNDED DEGREE SPANNING TREE problem is a special case when  $k = 1$ . Theorem 2 implies the following result, which is optimal in terms of the degree upper bounds.

**Corollary 3.** *There exists a polynomial time algorithm for the MINIMUM BOUNDED DEGREE SPANNING TREE UNION problem which returns a subgraph  $G$  of cost at most  $\text{OPT}$  which is the union of  $k$  edge-disjoint spanning trees and the degree of  $v$  in  $H$  is at most  $g(v) + 1$ . Here  $\text{OPT}$  is the cost of an optimal solution which satisfies all the degree upper bounds.*

*Proof.* Let  $M = (E, \mathcal{I})$  denote the union of  $k$  graphic matroids over the graph  $G$ , which is a matroid by the matroid union theorem. The hypergraph  $H$  is defined with  $V(H) = E(G)$  and  $E(H) = \{\delta(v) : v \in V(G)\}$ . Note that  $\Delta = \max_{v \in V(H)} |\{e \in E(H) : v \in e\}| = \max_{e \in E(G)} |\{\delta(v) : v \in V(G) \wedge e \in \delta(v)\}| = 2$ . So, using Theorem 2, we obtain a basis  $T$  of matroid  $M$  (which is the union of  $k$  edge-disjoint spanning trees), such that  $|T \cap C_i| \leq g_i + 1$ . □

### 2.4 Minimum Bounded Degree Directed Cut Cover

Let  $D = (V, E)$  be a digraph. A set of vertices  $X$  is called a *directed cut* if  $\delta^{out}(X) = \emptyset$ . A subset of edges  $F$  is called a *directed cut cover* if  $|F \cap \delta(X)| \neq \emptyset$  for every directed cut  $X$ . In the MINIMUM BOUNDED DEGREE DIRECTED CUT COVER problem, we are given a digraph  $D = (V, E)$ , a cost function  $c : E \rightarrow \mathbb{Z}$ , and degree constraints  $f(v)$  and  $g(v)$  for each  $v \in V$ . The task is to find a directed cut cover  $F \subseteq E$  of minimum cost such that  $f(v) \leq |F \cap \delta(v)| \leq g(v)$  for every  $v \in V$ . Theorem 3 implies the following result, which is optimal in terms of the degree bounds.

**Corollary 4.** *There exists a polynomial time algorithm for the MINIMUM BOUNDED DEGREE DIRECTED CUT COVER problem which returns a directed cut cover  $F$  of cost at most  $\text{OPT}$  and  $f(v) - 1 \leq |F \cap \delta(v)| \leq g(v) + 1$  for each vertex  $v \in V$ . Here  $\text{OPT}$  is the cost of an optimal solution which satisfies all the degree constraints.*

*Proof.* Let  $b(X) = -1$  if  $V \setminus X$  is a directed cut, and let  $b(X) = \infty$  otherwise. Then  $b$  is a crossing submodular set function. In this setting, a 0-1 submodular flow corresponds to a directed cut cover. So, by Theorem 3, we obtain a directed cut cover  $F$  such that  $f(v) - 1 \leq |F \cap \delta(v)| \leq g(v) + 1$  for every  $v \in V$ . □

### 2.5 Minimum Bounded Degree Graph Orientation

In the MINIMUM BOUNDED DEGREE GRAPH ORIENTATION problem, we are given a digraph  $D = (V, E)$ , a cost function  $c : E \rightarrow \mathbb{Z}$ , and bounds  $f(v) \leq g(v)$  for every  $v \in V$ . The task is to find an edge set of minimum cost whose reversal makes the digraph  $k$ -edge-connected, so that the number of edges reversed at each node  $v$  is between  $f(v)$  and  $g(v)$ . Theorem 3 implies the following result, which is optimal in terms of the degree bounds.

**Corollary 5.** *There exists a polynomial time algorithm for the MINIMUM BOUNDED DEGREE GRAPH ORIENTATION problem which finds an edge set of cost at most OPT whose reversal makes the digraph  $k$ -edge-connected and such that the number of edges reversed at each node  $v$  is between  $f(v) - 1$  and  $g(v) + 1$ . Here OPT is the cost of an optimal solution which satisfies all the degree constraints.*

*Proof.* This can be done by considering the submodular flow problem defined by the set function  $b(X) = |\delta^{in}(X)| - k$  ( $\emptyset \neq X \subsetneq V$ ) (see [6]), which is a submodular set function. In this setting, a 0-1 submodular flow corresponds to an edge set whose reversal makes the digraph strongly  $k$ -edge-connected. So this result follows from Theorem 3. □

It is shown in subsection 4.1 that the corresponding feasibility problem is NP-complete, and thus the feasibility problem for bounded degree submodular flow is also NP-complete.

### 3 Minimum Bounded Degree Matroid Basis

**Proof of Theorem 1:** The main technique used to prove Theorem 1 is the iterative relaxation method used in [10,12], which is based on the iterative rounding method introduced by Jain [9]. We first formulate a linear programming relaxation for the MINIMUM DEGREE BOUNDED MATROID BASIS problem. Let  $r : 2^V \rightarrow \mathbb{Z}_+$  denote the rank function of matroid  $M$ .

$$\text{minimize} \quad c(x) = \sum_{v \in V} c_v x_v \tag{4}$$

$$\text{subject to} \quad x(V) = r(V) \tag{5}$$

$$x(S) \leq r(S) \quad \forall S \subseteq V \tag{6}$$

$$f(e) \leq x(e) \leq g(e) \quad \forall e \in E(H) \tag{7}$$

$$0 \leq x_v \leq 1 \quad \forall v \in V \tag{8}$$

This linear program is exponential in size but can be separated over in polynomial time if given an access to the independent set oracle [4]. Given a matroid  $M = (V, \mathcal{I})$  and an element  $v \in V$ , we denote by  $M \setminus v$  the matroid obtained by deleting  $v$ , i.e.,  $M \setminus v = (V', \mathcal{I}')$  where  $V' = V \setminus \{v\}$  and  $\mathcal{I}' = \{S \in \mathcal{I} : v \notin S\}$ . We also denote by  $M/v$  the matroid obtained by contracting  $v$ , i.e.,  $M/v = (V', \mathcal{I}')$  where  $V' = V \setminus \{v\}$  and  $\mathcal{I}' = \{S \setminus \{v\} : S \in \mathcal{I}, v \in S\}$ .

The algorithm is given in Figure 1. Suppose that the algorithm terminates successfully. Then Theorem 1 follows from a similar argument as in [12], which is sketched as follows. Firstly, observe that the matroid  $M$  is updated to  $M \setminus v$  whenever we remove  $v$  such that  $x_v = 0$  and updated to  $M/v$  whenever we pick  $v$  such that  $x_v = 1$ . A simple verification shows that the residual linear programming solution (current LP solution restricted to  $V \setminus \{v\}$ ) remains a feasible solution for the modified linear program in the the next iteration. In Step 2 we remove a degree constraint, and hence the current linear programming solution

1. Initialization  $B \leftarrow \emptyset$ ,
2. While  $B$  is not a basis do
  - (a) Find a basic optimal solution  $x$ . Delete  $v$  such that  $x_v = 0$ . Update each edge  $e \in E(H)$  let  $e \leftarrow e \setminus \{v\}$ . Update matroid  $M \leftarrow M \setminus v$ .
  - (b) For each element  $v$  with  $x_v = 1$ , include  $v$  in  $B$  and decrease  $f(e)$  and  $g(e)$  by 1 for each  $e \ni v$ . Update matroid  $M \leftarrow M/v$ .
  - (c) For every  $e \in E(H)$  such that  $|e| \leq 2\Delta$ , remove  $e$  from  $E(H)$ .
3. Return  $B$ .

**Fig. 1.** The algorithm for the MINIMUM BOUNDED DEGREE MATROID BASIS problem

remains a feasible solution. So, a simple inductive argument shows that by only picking elements with  $x_v = 1$ , the cost of the returned basis is no more than the cost of the original basic optimal solution. Also, since we only remove a degree constraint of a hyperedge when it contains at most  $2\Delta$  elements, the degree constraints are violated by at most  $2\Delta - 1$ . Therefore, it remains to show that the algorithm always terminates successfully. That is, it can always find an element  $v$  with  $x_v = 1$  in Step 2b or it finds a hyperedge  $e$  with  $|e| \leq 2\Delta$  in Step 2c.

Suppose for contradiction neither of the above conditions hold. Hence,  $0 < x_v < 1$  for each  $v \in V$  and  $|e| > 2\Delta$  for each  $e \in E(H)$ . Let  $\mathcal{T} = \{S \subseteq V : x(S) = r(S)\}$  be the collection of all tight sets at solution  $x$ . Let  $\chi_S$  denote the characteristic vector of  $S$ , i.e.,  $\chi_S(v) = 1$  if  $v \in S$  else  $\chi_S(v) = 0$ . A family of sets  $\mathcal{L} \subseteq 2^V$  is called a *chain* if the following condition holds: for every  $A, B \in \mathcal{L}$  we have either  $A \subset B$  or  $B \subset A$ . The following claim can be obtained by standard uncrossing argument (see Schrijver [41] Chapter 41).

*Claim.* For any basic solution  $x$ , there exists a chain  $\mathcal{L} \subseteq \mathcal{T}$  such that the following holds.

1.  $\{\chi_S : S \in \mathcal{L}\}$  are linearly independent vectors.
2.  $\text{span}(\{\chi_S : S \in \mathcal{L}\}) = \text{span}(\{\chi_S : S \in \mathcal{T}\})$ .

As  $x$  is a basic solution, there is a set  $E' \subseteq E$  of tight hyperedges (a hyperedge  $e$  is tight if  $x(e) = g(e)$  or  $x(e) = f(e)$ ) such that  $\{\chi_S : S \in \mathcal{L}\} \cup \{\chi_e : e \in E'\}$  are linearly independent vectors and  $|V| = |E'| + |\mathcal{L}|$ . We now derive a contradiction to this by a counting argument. We assign  $2\Delta$  tokens to each vertex  $v \in V$  for a total of  $2\Delta|V|$  tokens. We then redistribute the tokens so that each hyperedge in  $E'$  collects at least  $2\Delta$  tokens, each member of  $\mathcal{L}$  collects at least  $2\Delta$  tokens, and there are still at least one extra token. This implies that  $2\Delta|V| > 2\Delta|E'| + 2\Delta|\mathcal{L}|$ , which gives us the desired contradiction.

The reassignment is as follows. Each element  $v$  gives  $\Delta$  tokens to the smallest member in  $\mathcal{L}$  it is contained in and one token to each hyperedge  $e \in E'$  it is present in. As any element is contained in at most  $\Delta$  edges, thus the redistribution is valid as we distribute at most  $2\Delta$  tokens per element. Now, consider any set  $S \in \mathcal{L}$  and let  $R$  be the largest set in  $\mathcal{L}$  contained in  $S$ . We have  $x(S) = r(S)$



and  $x(R) = r(R)$ . Thus, we have  $x(S \setminus R) = r(S) - r(R)$ . As constraints for  $R$  and  $S$  are linearly independent and  $x_v > 0$  for each  $v \in V$ , this implies  $r(S) \neq r(R)$ . Since  $r$  is a matroid rank function,  $r(S) - r(R) \geq 1$  as they are both integers. Since  $0 < x_v < 1$ , this implies  $|S \setminus R| \geq 2$ . Thus,  $S$  can collect at least  $2\Delta$  tokens,  $\Delta$  tokens from each element in  $S \setminus R$ , as required. Consider any hyperedge  $e \in E'$ . As  $|e| \geq 2\Delta$  and it can collect one token from each element in  $e$ , there are at least  $2\Delta$  tokens for each edge  $e$ , as required.

Now, it remains to argue that there is an extra token left. If any of the elements is in strictly less than  $\Delta$  hyperedges of  $E'$  then we have one extra token. Else,  $\sum_{e \in E'} \chi_e = \Delta \cdot \chi_V$  which gives dependence among the constraints as  $V \in \mathcal{L}$ . Hence, we have the desired contradiction, and the proof of Theorem 1 follows.  $\square$

Now we show how to use the proof technique of Bansal et al. 2 to obtain Theorem 2.

**Proof of Theorem 2:** The proof for upper bounds is similar to the proof of Theorem 1 except for the counting argument. The only important difference is that we remove a hyperedge  $e$  if  $g(e) + \Delta - 1 \geq |e|$ ; this is possible since in that case the degree upper bound on  $e$  can be violated by at most  $\Delta - 1$ . It follows that we may assume that  $|e| - g(e) \geq \Delta$  for all hyperedges.

The proof that  $|V| > |E'| + |\mathcal{L}|$  if  $0 < x < 1$  goes as follows. Let  $\mathcal{L} = \{S_1, \dots, S_k\}$ , where  $S_1 \subsetneq S_2 \subsetneq \dots \subsetneq S_k$ , and let  $S_0 := \emptyset$ . Then  $|e| - x(e) \geq \Delta$  for every  $e \in E'$ , and  $x(S_i \setminus S_{i-1}) = r(S_i) - r(S_{i-1}) \geq 1$  for  $i = 1, \dots, k$ . Using these inequalities, we obtain that

$$\begin{aligned} |E'| + |\mathcal{L}'| &\leq \sum_{e \in E'} \frac{|e| - x(e)}{\Delta} + \sum_{i=1}^k x(S_i \setminus S_{i-1}) \\ &= \sum_{v \in V} \frac{1 - x(v)}{\Delta} |\{e \in E' : v \in e\}| + x(S_k) \leq |V|, \end{aligned}$$

and if equality holds, then  $|\{e \in E' : v \in e\}| = \Delta$  for every  $v \in V$  and  $S_k = V$ . But then  $\Delta \cdot \chi_{S_k} = \sum_{e \in E'} \chi_e$ , which contradicts the linear independence.

If only lower bounds are present, then we can delete a hyperedge  $e$  if  $f(e) \leq \Delta - 1$ , so we may assume that  $f(e) \geq \Delta$  for all hyperedges. To show  $|V| > |E'| + |\mathcal{L}|$  we use that  $x(e) \geq \Delta$  for every  $e \in E'$  and  $|S_i \setminus S_{i-1}| - x(S_i \setminus S_{i-1}) \geq 1$  for  $i = 1, \dots, k$ , where the latter holds because  $x(S_i \setminus S_{i-1}) < |S_i \setminus S_{i-1}|$  and both are integer. Thus

$$\begin{aligned} |E'| + |\mathcal{L}'| &\leq \sum_{e \in E'} \frac{x(e)}{\Delta} + \sum_{i=1}^k (|S_i \setminus S_{i-1}| - x(S_i \setminus S_{i-1})) \\ &= \sum_{v \in V} \frac{x(v)}{\Delta} |\{e \in E' : v \in e\}| + |S_k| - x(S_k) \leq |V|, \end{aligned}$$

and the claim follows similarly as for upper bounds.  $\square$



*Remark 1.* It is shown in [12] that for the MINIMUM BOUNDED DEGREE SPANNING TREE problem the violation of the degree bounds can be bounded by  $\Delta - 1$  (which is equal to 1 since  $\Delta = 2$  in that problem) even in the presence of both lower and upper bounds on the degrees. In the generalization for matroids, it seems that our method cannot guarantee a solution that violates the bounds by at most  $\Delta - 1$  if both lower and upper degree bounds are present. The reason is that there may be a basic solution with non-integer values, but Step 2c can not be applied, as the following example shows.

Let  $V = \{u_1, u_2, \dots, u_6, v_1, v_2, \dots, v_6\}$  be a ground set of 12 elements, and let  $M = (V, \mathcal{I})$  be the partition matroid where each basis contains 1 element from each of  $\{u_1, v_1\}$ ,  $\{u_3, v_3\}$ ,  $\{u_4, v_2\}$ , and  $\{u_6, v_5\}$ , and 2 elements from  $\{u_2, u_5, v_4, v_6\}$ . Let  $H = (V, E)$  be the hypergraph containing the hyperedges  $\{u_1, u_2, u_3\}$ ,  $\{u_3, u_4, u_5\}$ ,  $\{u_5, u_6, u_1\}$ ,  $\{u_2, u_4, u_6\}$ , and  $\{v_1, v_2, v_3\}$ ,  $\{v_3, v_4, v_5\}$ ,  $\{v_5, v_6, v_1\}$ ,  $\{v_2, v_4, v_6\}$ . For the first four hyperedges, let the lower bound  $f(e)$  be 2, and for the last four hyperedges, let the upper bound  $g(e)$  be 1. Then the following is a basic solution:  $u_i = 2/3$  ( $i = 1, \dots, 6$ ),  $v_i = 1/3$  ( $i = 1, \dots, 6$ ). It is not possible to delete any hyperedges since  $f(e) \geq \Delta$  or  $|e| - g(e) \geq \Delta$  for each hyperedge  $e \in E$ .

## 4 Minimum Bounded Degree Submodular Flow

**Proof of Theorem 3:** The proof of this theorem is also based on the iterative relaxation method used in [10,12]. Let us define the linear relaxation of the problem by

$$\text{minimize} \quad c(x) = \sum_{e \in E} c(e)x(e) \tag{9}$$

$$x(\delta^{in}(X)) - x(\delta^{out}(X)) \leq b(X) \quad \text{for every } X \subseteq V, \tag{10}$$

$$x(\delta(v)) \geq f(v) \quad \text{for every } v \in V_f, \tag{11}$$

$$x(\delta(v)) \leq g(v) \quad \text{for every } v \in V_g, \tag{12}$$

$$0 \leq x(e) \leq 1 \quad \text{for every } e \in E. \tag{13}$$

Let  $x^*$  be an optimal basic solution of the linear programming relaxation. This can be obtained in polynomial time by the ellipsoid method. Obviously  $c(x^*) \leq \text{OPT}$ . We will find a 0-1 submodular flow of cost at most  $c(x^*)$  that violates the degree bounds by at most one.

The problem can be reduced to an instance containing fewer edges in two cases:

- If  $x^*(e) = 0$  for some  $e \in E$ , then we delete the edge  $e$  from the digraph. A solution of the resulting problem solves the original problem.
- If  $x^*(e) = 1$  for some  $e = uv \in E$ , then we delete the edge  $e$  from the digraph, decrease  $f(u), f(v), g(u), g(v)$  by 1, and change  $b$  as follows:

$$b'(X) = \begin{cases} b(X) - 1 & \text{if } u \notin X \text{ and } v \in X, \\ b(X) + 1 & \text{if } u \in X \text{ and } v \notin X, \\ b(X) & \text{otherwise.} \end{cases}$$

The set function  $b'$  is also crossing submodular. If we have a solution  $x'$  for this modified problem, then we can obtain a solution for the original problem by setting  $x'(e) = 1$ .

This way we can reduce the problem to an instance where  $0 < x^*(e) < 1$  for every  $e \in E$ . We may also delete isolated nodes by changing  $b$  appropriately. Now we try to remove degree bounds so that the solutions of the resulting problem are feasible for the original problem. One difference from the proof of Theorem 1 is that in some iterations we increase the number of vertices in the graph. However, in each step we decrease  $|E| + |V_f| + |V_g|$  by at least one and thus the number of steps is polynomial.

First let us observe that  $g(v) > 0$  for every  $v \in V_g$  and  $f(v) < |\delta(v)|$  for every  $v \in V_f$ , since otherwise there would be some edge  $e$  with  $x^*(e) = 0$  or  $x^*(e) = 1$ . Removal of an upper degree bound at a node  $v$  is possible in the following two cases:

- If  $|\delta(v)| \leq g(v) + 1$ , then we can remove the upper bound at  $v$ , since a solution of the resulting problem cannot violate the original degree bound by more than 1.
- If  $g(v) = 1$ , then we replace  $v$  by two nodes  $v_1$  and  $v_2$ . An edge  $uv \in E$  is replaced by  $uv_1$ , while an edge  $vu \in E$  is replaced by  $v_2u$ . The set function  $b$  is modified as follows:

$$b'(X) = \begin{cases} 1 & \text{if } X = v_1 \text{ or } X = V - v_2, \\ b(X) & \text{if } X \cap \{v_1, v_2\} = \emptyset, \\ b(X - \{v_1, v_2\}) + v & \text{if } \{v_1, v_2\} \subseteq X, \\ \infty & \text{otherwise.} \end{cases}$$

The set function  $b'$  is crossing submodular. No degree upper bound and lower bound are given for  $v_1$  and  $v_2$ , i.e.  $V'_g = V_g - v, V'_f = V_f - v$ . Note that the current solution corresponds to a feasible solution of this relaxation. The definition of  $b'$  implies that  $x(\delta(v_1)) \leq 1$  and  $x(\delta(v_2)) \leq 1$  for any solution  $x$ . This means that the corresponding solution on the original digraph violates the degree bounds at  $v$  by at most 1.

After the above modifications, we may assume that  $g(v) \geq 2$  and  $|\delta(v)| \geq g(v) + 2$  for every  $v \in V_g$ . Removal of a lower degree bound at a node  $v$  is possible in the following two cases:

- If  $f(v) \leq 1$ , then we can remove the lower bound at  $v$ , since a solution of the resulting problem cannot violate the original bound by more than 1.
- If  $f(v) = 2$  and  $|\delta(v)| = 3$ , then we replace  $v$  by two nodes  $v_1$  and  $v_2$ . An edge  $uv \in E$  is replaced by  $uv_1$ , while an edge  $vu \in E$  is replaced by  $v_2u$ . For

the modification of  $b$  there are two cases. If  $|\delta^{out}(v)| \leq 1$ , then it is modified as follows:

$$b'(X) = \begin{cases} -1 & \text{if } X = V - v_1, \\ b(X) & \text{if } X \cap \{v_1, v_2\} = \emptyset, \\ b(X - \{v_1, v_2\} + v) & \text{if } \{v_1, v_2\} \subseteq X, \\ \infty & \text{otherwise.} \end{cases}$$

If  $|\delta^{in}(v)| \leq 1$ , then the modified set function is

$$b'(X) = \begin{cases} -1 & \text{if } X = v_2, \\ b(X) & \text{if } X \cap \{v_1, v_2\} = \emptyset, \\ b(X - \{v_1, v_2\} + v) & \text{if } \{v_1, v_2\} \subseteq X, \\ \infty & \text{otherwise.} \end{cases}$$

The set function  $b'$  is crossing submodular. No lower bound is given for  $v_1$  and  $v_2$ , i.e.  $V'_f = V_f - v$ . Note that there is no degree upper bound on  $v$  by the previous rule (since  $g(v) \geq f(v) \geq |\delta(v)| - 1$ ), and the current solution corresponds to a feasible solution in this relaxation. The definition of  $b'$  implies that  $x(\delta(\{v_1, v_2\})) \geq 1$  for any solution  $x$ . This means that the corresponding solution on the original digraph violates the lower bound at  $v$  by at most 1.

After the above modifications, we may assume that  $|\delta(v)| \geq 4$  for every  $v \in V_f \cup V_g$ . The solution corresponding to  $x^*$  is still a feasible solution, but it is not necessarily a basic solution, so we have to resolve the LP and continue this process until a basic solution is obtained where there are no 0-1 edges and no degree bounds can be deleted. (Actually, it is not necessary to solve the LP to optimality, it is enough to perform the easier task of finding a basic solution that is not worse than the current solution).

At the end of the process either all edges are fixed to 0 or 1 and we are done, or  $0 < x^*(e) < 1$  for every  $e \in E$ , there are no isolated nodes, and  $|\delta(v)| \geq 4$  for every  $v \in V_f \cup V_g$ . We show that the latter case is impossible. Since  $x^*$  is a basic solution, there is a system of linearly independent constraints which are tight at  $x^*$  for which  $x^*$  is the unique solution of the equation system given by these tight constraints. Let  $\mathcal{F}^*$  be the family of sets corresponding to the submodular flow constraints in this system, and let  $V^*$  denote the set of nodes with degree constraints that are in the system. A family of sets  $\mathcal{F} \subset 2^V$  is called *cross-free* if for every sets  $A, B \in \mathcal{F}$  we have either  $A \subseteq B$ ,  $B \subseteq A$ ,  $A \cap B = \emptyset$  or  $A \cup B = V$ . The following claim can be obtained by standard uncrossing argument (see Schrijver [11] Chapter 60).

*Claim.* We may assume that the family  $\mathcal{F}^*$  is cross-free.

Since  $x^*$  is the unique solution of the equation system defined by  $\mathcal{F}^*$  and  $V^*$ , we have  $|E| \leq |\mathcal{F}^*| + |V^*|$ . We show, using a simple counting argument, that this is impossible.

We assign  $2|E|$  tokens to the nodes by assigning 2 tokens for every edge in  $E$  to the two endpoints of the edge. The idea of the proof is to reassign these tokens to the members of  $\mathcal{F}^*$  and  $V^*$  so that every member gets at least 2 tokens and at least one token is not assigned to any member. This would contradict  $|E| \leq |\mathcal{F}^*| + |V^*|$ .

Let  $r \in V$  be an arbitrary node. We define the family

$$\mathcal{H}^* := \{X \subseteq V - r : X \in \mathcal{F}^*\} \cup \{X \subseteq V - r : V - X \in \mathcal{F}^*\}.$$

Notice that  $\mathcal{H}^*$  is laminar. For a set  $X \in \mathcal{H}^*$ , we define  $X' \in \mathcal{F}^*$  to be either  $X$  or  $V - X$  (depending on which one is in  $\mathcal{F}^*$ ). We will assign 2 tokens to each member of  $\mathcal{H}^*$  so that every member gets tokens from its nodes, thus the tokens of  $r$  are not used.

A node  $v \in V^*$  has at least 4 tokens since  $|\delta(v)| \geq 4$ . We assign 2 of its tokens to  $v$  (as degree constraint) and 2 tokens to the smallest member of  $\mathcal{H}^*$  containing  $v$ . If no member of  $\mathcal{H}^*$  contains  $v$ , we have 2 unused tokens.

To assign tokens to the remaining members of  $\mathcal{H}^*$ , we proceed in an order compatible with the partial order of inclusion. Let  $X \in \mathcal{H}^*$  be a set that has no tokens yet, and let  $\{X_1, \dots, X_k\}$  be the maximal members of  $\mathcal{H}^*$  inside  $X$ , which all have at least two tokens assigned to them. There must be an edge with an endpoint in  $X - \cup_{i=1}^k X_i$ , otherwise the constraints corresponding to  $X', X'_1, \dots, X'_k$  would be linearly dependent: the constraint for  $X'$  would be a  $\pm 1$  combination of the constraints for  $X'_1, \dots, X'_k$ , where the  $i$ -th coefficient depends on whether  $X'_i = X_i$  or  $X'_i = V - X_i$ . Moreover, if only one such edge  $e$  existed, then  $x^*(e)$  would be integer because it would be determined by an integer combination  $b(X'), b(X'_1), \dots, b(X'_k)$ . Since  $0 < x^*(e) < 1$  for every edge, it follows that there are at least two edges with an endpoint in  $X - \cup_{i=1}^k X_i$ , hence there are at least two tokens inside  $X$  that are not yet assigned to other sets. We assign these tokens to  $X$ .

At the end of this procedure, every member of  $\mathcal{H}^*$  and  $V^*$  is assigned 2 tokens, and there is an unused token at  $r$  since it is not an isolated node. This contradicts the assumption that  $|E| \leq |\mathcal{F}^*| + |V^*|$ , so we proved the theorem.  $\square$

### 4.1 Hardness of the Feasibility Problem

In this section we prove that a special case of the degree-constrained 0-1 submodular flow problem is NP-complete. The construction also shows that the feasibility problems for BOUNDED DEGREE GRAPH ORIENTATION and BOUNDED DEGREE DIRECTED CUT COVER are NP-complete. A subset of edges in a digraph is called *independent* if no two edges have a common node. In the following  $E[W]$  denotes the set of induced edges in  $W$ , i.e. edges with both endpoints in  $W$ .

**Theorem 4.** *Given a digraph  $D = (V, E)$  and a subset  $W \subseteq V$  of nodes, it is NP-complete to decide if it is possible to change the orientation of an independent subset of edges in  $E[W]$  so that the resulting digraph is strongly connected.*

*Proof.* We reduce SAT to this problem. Let us consider a SAT instance with variables  $x_1, \dots, x_n$  and clauses  $c_1, \dots, c_m$ . We associate a digraph  $D = (V, E)$  and a node set  $W \subseteq V$  to this instance using the following construction.

For the variable  $x_j$ , let  $m_j$  be the number of clauses that contain  $x_j$  or  $\neg x_j$ . We construct a cycle of length  $4m_j$ : the nodes are  $u_i^j, v_i^j, w_i^j, z_i^j$  ( $i = 1, \dots, m_j$ ), the oriented edges are  $u_i^j v_i^j, w_i^j v_i^j, z_i^j w_i^j, z_i^j u_{i+1}^j$  ( $i = 1, \dots, m_j$ ). The node set  $W$  consists of all these nodes.

In addition, we add a node  $t$  and nodes  $s_i$  ( $i = 1, \dots, m$ ), and add edges  $s_i t$  ( $i = 1, \dots, m$ ). For a given variable  $x_j$ , suppose that  $c_i$  is the  $i$ -th clause that contains  $x_j$  or  $\neg x_j$ . If it contains  $x_j$ , then we add the edges  $s_i u_i^j, u_i^j s_i, w_i^j t, t w_i^j$ . If it contains  $\neg x_j$ , then we add the edges  $s_i w_i^j, w_i^j s_i, u_i^j t, t u_i^j$ . This finishes the construction of the digraph  $D$ .

Consider the cycle of length  $4m_j$  associated to the variable  $x_j$ . The nodes  $v_i^j$  have out-degree 0, while the nodes  $z_i^j$  have in-degree 0 ( $i = 1, \dots, m_j$ ). This means that we have to change the orientation of  $2m_j$  independent edges in the cycle in order to get a strong orientation. Thus we have two possibilities: either we change the orientation of the edges  $u_i^j v_i^j, z_i^j w_i^j$  ( $i = 1, \dots, m_j$ ), or of the edges  $w_i^j v_i^j, z_i^j u_{i+1}^j$  ( $i = 1, \dots, m_j$ ). We say that the former corresponds to the ‘true’ value of  $x_j$ , while the later corresponds to the ‘false’ value.

In this way, there is a one-to-one correspondence between orientations of the above structure and possible evaluations of the variables. We claim that the orientation is strongly connected if and only if the corresponding evaluation satisfies the SAT formula. Suppose that the formula is not satisfied, i.e. there is a clause  $c_i$  containing only false literals. Consider the node set consisting of  $s_i$  and its neighbors of type  $u$  and  $w$ . By the construction, this set has in-degree 0 in the orientation corresponding to the evaluation. Therefore the orientation cannot be strongly connected.

Now suppose that an evaluation satisfies the formula. Then each node  $s_i$  ( $i = 1, \dots, m$ ) can be reached from  $t$  by a path of length 4 (which corresponds to the “true” literal in  $c_i$ ). Since there is an edge from  $s_i$  to  $t$  for each  $s_i$ , and all other nodes obviously have paths to and from  $t$  or some  $s_i$ , the orientation is strongly connected.  $\square$

**Corollary 6.** *The feasibility problem for degree-constrained 0-1 submodular flows is NP-complete.*

## References

1. Bansal, N., Khandekar, R., Nagarajan, V.: Additive Guarantees for Degree Bounded Directed Network Design, IBM Research Report RC24347 (September 2007)
2. Bilo, V., Goyal, V., Ravi, R., Singh, M.: On the Crossing Spanning Tree Problem. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) RANDOM 2004 and APPROX 2004. LNCS, vol. 3122, pp. 51–60. Springer, Heidelberg (2004)
3. Chaudhuri, K., Rao, S., Riesenfeld, S., Talwar, K.: A Push-Relabel Algorithm for Approximating the Minimum-Degree MST Problem and its Generalization to Matroids, Invited submission to Theoretical Computer Science (Special Issue for ICALP 2006) (2006)

4. Cunningham, W.H.: Testing membership in matroid polyhedra. *Journal of Combinatorial Theory, Series B* 36(2), 161–188 (1984)
5. Edmonds, J., Giles, R.: A min-max relation for submodular functions on graphs. *Ann. Discrete Math.* 1, 185–204 (1977)
6. Frank, A.: An algorithm for submodular functions on graphs. *Ann. Discrete Math.* 16, 97–120 (1982)
7. Frieze, A.: Personal Communication (March 2007)
8. Goemans, M.X.: Minimum bounded-degree spanning trees. In: *Proceedings of 47th IEEE FOCS*, pp. 273–282 (2006)
9. Jain, K.: A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica* 21, 39–60 (2001)
10. Lau, L.C., Naor, J., Salavatipour, M., Singh, M.: Survivable network design with degree or order constraints. In: *Proceedings of 39th ACM STOC*, pp. 651–660 (2007)
11. Schrijver, A.: *Combinatorial Optimization, Polyhedra and Efficiency*. Springer, Heidelberg (2003)
12. Singh, M., Lau, L.C.: Approximating minimum bounded degree spanning trees to within one of optimal. In: *Proceedings of the 39th ACM STOC*, pp. 661–670 (2007)

# Budgeted Matching and Budgeted Matroid Intersection Via the Gasoline Puzzle<sup>\*</sup>

André Berger<sup>1</sup>, Vincenzo Bonifaci<sup>2,3,\*\*</sup>,  
Fabrizio Grandoni<sup>4,\*\*\*</sup>, and Guido Schäfer<sup>5</sup>

<sup>1</sup> Department of Quantitative Economics, University of Maastricht, The Netherlands  
a.berger@ke.unimaas.nl

<sup>2</sup> Department of Electrical Engineering, University of L'Aquila, Italy

<sup>3</sup> Department of Computer and Systems Science, Sapienza University of Rome, Italy  
bonifaci@dis.uniroma1.it

<sup>4</sup> Department of Computer Science, Systems and Production,  
University of Rome Tor Vergata, Italy  
grandoni@disp.uniroma2.it

<sup>5</sup> Institute for Mathematics, Technical University Berlin, Germany  
schaefer@math.tu-berlin.de

**Abstract.** Many polynomial-time solvable combinatorial optimization problems become NP-hard if an additional complicating constraint is added to restrict the set of feasible solutions. In this paper, we consider two such problems, namely maximum-weight matching and maximum-weight matroid intersection with one additional budget constraint. We present the first polynomial-time approximation schemes for these problems. Similarly to other approaches for related problems, our schemes compute two solutions to the Lagrangian relaxation of the problem and patch them together to obtain a near-optimal solution. However, due to the richer combinatorial structure of the problems considered here, standard patching techniques do not apply. To circumvent this problem, we crucially exploit the adjacency relations on the solution polytope and, surprisingly, the solution to an old combinatorial puzzle.

## 1 Introduction

Many combinatorial optimization problems can be formulated as follows. We are given a (finite) set  $\mathcal{F}$  of feasible solutions and a weight function  $w : \mathcal{F} \rightarrow \mathbb{Q}$  that assigns a weight  $w(S)$  to every feasible solution  $S \in \mathcal{F}$ . An *optimization problem*  $\Pi$  asks for the computation of a feasible solution  $S^* \in \mathcal{F}$  of maximum weight  $opt_{\Pi}$ , i.e.,

$$opt_{\Pi} := \text{maximize } w(S) \quad \text{subject to } S \in \mathcal{F}. \quad (\Pi)$$

---

<sup>\*</sup> This work was done while the first three authors were postdoctoral fellows at TU Berlin. Research supported by the European Regional Development Fund (ERDF).

<sup>\*\*</sup> This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

<sup>\*\*\*</sup> Research supported by MIUR under project MAINSTREAM.

In this paper, we are interested in solving such optimization problems if the set of feasible solutions is further constrained by a single *budget constraint*. More precisely, we are additionally given a non-negative cost function  $c : \mathcal{F} \rightarrow \mathbb{Q}^+$  that specifies a cost  $c(S)$  for every feasible solution  $S \in \mathcal{F}$  and a non-negative budget  $B \in \mathbb{Q}^+$ . The *budgeted optimization problem*  $\bar{\Pi}$  of the above problem  $\Pi$  can then be formulated as follows:

$$\text{opt} := \text{maximize } w(S) \text{ subject to } S \in \mathcal{F}, c(S) \leq B. \tag{\bar{\Pi}}$$

Even if the original optimization problem  $\Pi$  is polynomial-time solvable, adding a budget constraint typically renders the budgeted optimization problem  $\bar{\Pi}$  NP-hard. Problems that fall into this class are, for example, the constrained shortest path problem [2], the constrained minimum spanning tree problem [1], and the constrained minimum arborescence problem [6].

We study the budgeted version of two fundamental optimization problems, namely the maximum-weight matching problem and the maximum-weight matroid intersection problem:

In the *budgeted matching problem*, we are given an undirected graph  $G = (V, E)$  with edge weights  $w : E \rightarrow \mathbb{Q}$  and edge costs  $c : E \rightarrow \mathbb{Q}^+$ , and a budget  $B \in \mathbb{Q}^+$ . The set  $\mathcal{F}$  of feasible solutions corresponds to the set of all matchings in  $G$ . Define the weight of a matching  $M$  as the total weight of all edges in  $M$ , i.e.,  $w(M) := \sum_{e \in M} w(e)$ . Similarly, the cost of  $M$  is defined as  $c(M) := \sum_{e \in M} c(e)$ . The goal is to compute a matching  $M^* \in \mathcal{F}$  of maximum weight  $w(M^*)$  among all matchings  $M$  in  $\mathcal{F}$  whose cost  $c(M)$  is at most  $B$ .

In the *budgeted matroid intersection problem*, we are given two matroids  $\mathcal{M}_1 = (E, \mathcal{F}_1)$  and  $\mathcal{M}_2 = (E, \mathcal{F}_2)$  on a common ground set of elements  $E$  (formal definitions will be given in Section 2). Moreover, we are given element weights  $w : E \rightarrow \mathbb{Q}$ , element costs  $c : E \rightarrow \mathbb{Q}^+$ , and a budget  $B \in \mathbb{Q}^+$ . The set of all feasible solutions  $\mathcal{F} := \mathcal{F}_1 \cap \mathcal{F}_2$  is defined by the intersection of  $\mathcal{M}_1$  and  $\mathcal{M}_2$ . The weight of an independent set  $X \in \mathcal{F}$  is defined as  $w(X) := \sum_{e \in X} w(e)$  and the cost of  $X$  is  $c(X) := \sum_{e \in X} c(e)$ . The goal is to compute a common independent set  $X^* \in \mathcal{F}$  of maximum weight  $w(X^*)$  among all feasible solutions  $X \in \mathcal{F}$  satisfying  $c(X) \leq B$ . Problems that can be formulated as the intersection of two matroids are, for example, matchings in bipartite graphs, arborescences in directed graphs, spanning forests in undirected graphs, etc.

A special case of both budgeted matching and budgeted matroid intersection is the budgeted matching problem on bipartite graphs. This problem is NP-hard by a simple reduction from the knapsack problem. We remark that the unbudgeted versions of the two problems can be solved in polynomial-time (see, e.g., [20]).

*Our Contribution.* We give the first polynomial-time approximation schemes (PTAS) for the budgeted matching problem and the budgeted matroid intersection problem. For a given input parameter  $\epsilon > 0$ , our algorithms compute a  $(1 - \epsilon)$ -approximate solution in time  $O(m^{O(1/\epsilon)})$ , where  $m$  is the number of edges in the graph or the number of elements in the ground set, respectively.



The basic structure of our polynomial-time approximation schemes resembles similar approaches for related budgeted optimization problems [18]. By dualizing the budget constraint of  $\bar{I}$  and lifting it into the objective function, we obtain for any  $\lambda \geq 0$  the Lagrangian relaxation  $\text{LR}(\lambda)$ .

$$z(\lambda) := \text{maximize } (w(S) + \lambda(B - c(S))) \quad \text{subject to } S \in \mathcal{F}. \quad (\text{LR}(\lambda))$$

Note that the relaxed problem  $\text{LR}(\lambda)$  is equivalent to the optimization problem  $\Pi$  with modified *Lagrangian weights*  $w_\lambda(e) := w(e) - \lambda c(e)$  for all  $e \in E$ . Since the unbudgeted problem  $\Pi$  is polynomial-time solvable, we can compute the optimal *Lagrangian multiplier*  $\lambda^* := \arg \min_{\lambda \geq 0} z(\lambda)$  and two optimal solutions  $S_1$  and  $S_2$  to  $\text{LR}(\lambda^*)$  such that  $c(S_1) \leq B \leq c(S_2)$ . (Details will be given in Section 2.) The idea now is to *patch*  $S_1$  and  $S_2$  together to obtain a feasible solution  $S$  for  $\bar{I}$  whose weight  $w(S)$  is at least  $(1 - \epsilon)\text{opt}$ . Our patching consists of two phases: an *exchange phase* and an *augmentation phase*.

*Exchange Phase:* Consider the polytope induced by the feasible solutions  $\mathcal{F}$  to the unbudgeted problem  $\Pi$  and let  $F$  be the face given by the solutions of maximum Lagrangian weight. This face contains both  $S_1$  and  $S_2$ . In the first phase, we iteratively replace either  $S_1$  or  $S_2$  with another vertex on  $F$ , preserving the invariant  $c(S_1) \leq B \leq c(S_2)$ , until we end up with two adjacent solutions. Note that both solutions have objective value  $z(\lambda^*) \geq \text{opt}$ . However, with respect to their original weights, we can only infer that  $w(S_i) = z(\lambda^*) - \lambda^*(B - c(S_i))$ . That is, we cannot hope to use these solutions directly:  $S_1$  is a feasible solution for  $\bar{I}$  but its weight  $w(S_1)$  might be arbitrarily far from  $\text{opt}$ . In contrast,  $S_2$  has weight  $w(S_2) \geq \text{opt}$ , but is infeasible.

*Augmentation Phase:* In this phase, we exploit the properties of adjacent solutions in the solution polytope. For matchings it is known that two solutions are adjacent in the matching polytope if and only if their symmetric difference is an alternating cycle or path  $X$ . Analogously, two adjacent extreme points in the common basis polytope of two matroids can be characterized by a proper alternating cycle  $X$  in the corresponding exchangeability graph [4,9]. The idea is to patch  $S_1$  according to a proper subpath  $X'$  of  $X$ . This subpath  $X'$  guarantees that the Lagrangian weight of  $S_1$  does not decrease too much, while at the same time the gap between the budget and the cost of  $S_1$  (and hence also the gap between  $w(S_1)$  and  $z(\lambda^*)$ ) is reduced. This way we obtain a feasible solution  $S$  whose weight differs from  $\text{opt}$  by at most the weight of two edges (elements).

Of course, constructing such a solution  $S$  alone is not sufficient to obtain a PTAS. (The maximum weight of an edge (element) might be comparable to the weight of an optimum solution). However, this problem can be easily overcome by guessing the edges (elements) of largest weight in the optimum solution in a preliminary step.

Surprisingly, the key ingredient that enables us to prove that there always exists a good patching subpath stems from an old combinatorial puzzle which we quote from the book by Lovász [10, Problem 3.21]. We leave the proof as an exercise to the reader.

“Along a speed track there are some gas-stations. The total amount of gasoline available in them is equal to what our car (which has a very large tank) needs for going around the track. Prove that there is a gas-station such that if we start there with an empty tank, we shall be able to go around the track without running out of gasoline.”

*Related Work.* For the budgeted matching problem there is an optimal algorithm if the costs are uniform. This problem is equivalent to finding a maximum-weight matching that consists of at most  $B$  edges, which can be solved by a reduction to perfect matching. Not much is known for the budgeted matching problem with general edge costs, besides that it is NP-hard. Naor et al. [15] proposed a fully polynomial-time approximation scheme (FPTAS) for an even more general class of problems, which contains the budgeted matching problem considered here as special case. However, personal communication [14] revealed that unfortunately the stated result [15, Theorem 2.2] is incorrect. To the best of our knowledge, the budgeted version of the maximum-weight matroid intersection problem has not been considered before.

Budgeted versions of polynomial-time solvable optimization problems have been studied extensively. The best known ones are probably the constrained shortest path problem and the constrained minimum spanning tree problem. Finding a shortest  $s, t$ -path  $P$  (with respect to weight) between two vertices  $s$  and  $t$  in a directed graph with edge weights and edge costs such that the total cost of  $P$  is at most  $B$  appears as an NP-hard problem already in the book by Garey and Johnson [5]. Similarly, finding a minimum weight spanning tree whose total cost is at most some specified value is NP-hard as well [1].

Goemans and Ravi [18] obtain a PTAS for the constrained minimum spanning tree problem by using an approach which resembles our exchange phase. Starting from two spanning trees obtained from the Lagrangian relaxation, they walk along the optimal face (with respect to the Lagrangian weights) of the spanning tree polytope until they end up with two adjacent solutions  $S_1$  and  $S_2$  with  $c(S_1) \leq B \leq c(S_2)$ . In this polytope, two spanning trees are adjacent if and only if their symmetric difference consists of just two edges. Therefore, the final solution  $S_1$  is a feasible spanning tree whose weight is away from the optimum by the weight of only one edge. In particular, once two such adjacent solutions have been found there is no need for an additional augmentation phase, which is instead crucial for matchings and matroid intersections. The PTAS by Goemans and Ravi [18] also extends to the problem of finding a minimum-weight basis in a matroid subject to a budget constraint.

Hassin and Levin [7] later improved the result of Goemans and Ravi and obtained an EPTAS for the constrained minimum spanning tree problem. A fully polynomial bicriteria approximation scheme for the problem has been found by Hong et al. [8]. However, the question whether there exists a fully polynomial-time approximation scheme to the constrained minimum spanning tree problem is open.

Finding constrained minimum arborescences in directed graphs is NP-hard as well. Guignard and Rosenwein [6] apply Lagrangian relaxation to solve it to optimality (though not in polynomial time). Previous work on budgeted optimization problems also includes results on budgeted scheduling [21] and bicriteria results for several budgeted network design problems [11].

All problems mentioned above can be interpreted as bicriteria optimization problems with a min-min objective, i.e., where the goal is to compute a solution that minimizes the objective value and whose cost stays below a given budget. In contrast, in our work we consider max-min bicriteria problems.

*Organization of the Paper.* The paper is structured as follows. In Section 2, we give some prerequisites on matroids and Lagrangian relaxation. We then present the PTAS for the budgeted matching problem in Section 3. The PTAS for the budgeted matroid intersection problem is the subject of Section 4. In Section 5 we discuss some open problems.

## 2 Preliminaries

### 2.1 Matroids

Let  $E$  be a set of elements and  $\mathcal{F} \subseteq 2^E$  be a non-empty set of subsets of  $E$ . Then  $\mathcal{M} = (E, \mathcal{F})$  is a *matroid* if the following holds:

- (a) If  $I \in \mathcal{F}$  and  $J \subseteq I$ , then  $J \in \mathcal{F}$ .
- (b) For every  $I, J \in \mathcal{F}$ ,  $|I| = |J|$ , for every  $x \in I$  there is a  $y \in J$  such that  $I \setminus \{x\} \cup \{y\} \in \mathcal{F}$ .

The elements of  $\mathcal{F}$  are called *independent sets*. An independent set  $X$  is a *basis* of  $\mathcal{M}$  if for every  $x \in E \setminus X$ ,  $X \cup \{x\} \notin \mathcal{F}$ . We assume that  $\mathcal{F}$  is represented implicitly by an oracle: for any given  $I \subseteq E$ , this oracle determines whether  $I \in \mathcal{F}$  or not. In the running time analysis, each query to the oracle is assumed to take constant time. It is not hard to show that matroids have the following properties (see e.g. [20] and references therein).

**Lemma 1.** *For any given matroid  $\mathcal{M} = (E, \mathcal{F})$ :*

1. (*Deletion*) For every  $E_0 \subseteq E$ ,  $\mathcal{M} - E_0 := (E', \mathcal{F}')$  is a matroid, where  $E' := E \setminus E_0$  and  $\mathcal{F}' := \{X \in \mathcal{F} : X \cap E_0 = \emptyset\}$ .
2. (*Contraction*) For every  $E_0 \in \mathcal{F}$ ,  $\mathcal{M}/E_0 := (E', \mathcal{F}')$  is a matroid, where  $E' := E \setminus E_0$  and  $\mathcal{F}' := \{X \subseteq E \setminus E_0 : X \cup E_0 \in \mathcal{F}\}$ .
3. (*Truncation*) For every  $q \in \mathbb{N}$ ,  $\mathcal{M}^q := (E, \mathcal{F}^q)$  is a matroid, where  $\mathcal{F}^q := \{X \in \mathcal{F} : |X| \leq q\}$ .
4. (*Extension*) For every set  $D$ ,  $D \cap E = \emptyset$ ,  $\mathcal{M} + D := (E', \mathcal{F}')$  is a matroid, where  $E' := E \cup D$  and  $\mathcal{F}' := \{X \subseteq E \cup D : X \cap E \in \mathcal{F}\}$ .

Observe that an oracle for the original matroid implicitly defines an oracle for all the derived matroids above. Given  $X \in \mathcal{F}$  and  $Y \subseteq E$ , the *exchangeability graph* of  $\mathcal{M}$  with respect to  $X$  and  $Y$  is the bipartite graph  $\text{ex}_{\mathcal{M}}(X, Y) := (X \setminus Y, Y \setminus X; H)$  with edge set  $H = \{(x, y) : x \in X \setminus Y, y \in Y \setminus X, X \setminus \{x\} \cup \{y\} \in \mathcal{F}\}$ .

**Lemma 2** ([9]). (*Exchangeability Lemma*) Given  $X \in \mathcal{F}$  and  $Y \subseteq E$ , if  $\text{ex}_{\mathcal{M}}(X, Y)$  has a unique perfect matching, then  $Y \in \mathcal{F}$ .

The *intersection* of two matroids  $\mathcal{M}_1 = (E, \mathcal{F}_1)$  and  $\mathcal{M}_2 = (E, \mathcal{F}_2)$  over the same ground set  $E$  is the pair  $\mathcal{M} = (E, \mathcal{F}_1 \cap \mathcal{F}_2)$ . We remark that the intersection of two matroids might not be a matroid, while every matroid  $\mathcal{M} = (E, \mathcal{F})$  is the intersection of itself with the trivial matroid  $(E, 2^E)$ . Lemma 1 can be naturally extended to matroid intersections. For example, for a given matroid intersection  $(E, \mathcal{F}_1 \cap \mathcal{F}_2)$ , by Lemma 1.3  $(E, \mathcal{F}_1^q \cap \mathcal{F}_2^q)$  is still the intersection of two matroids, for any  $q \in \mathbb{N}$ .

Given two matroids  $\mathcal{M}_1 = (E, \mathcal{F}_1)$  and  $\mathcal{M}_2 = (E, \mathcal{F}_2)$ , the *common basis polytope* of  $\mathcal{M}_1$  and  $\mathcal{M}_2$  is the convex hull of the characteristic vectors of the common bases. We say that two common bases  $X, Y \in \mathcal{F}_1 \cap \mathcal{F}_2$  are *adjacent* if their characteristic vectors are adjacent extreme points in the common basis polytope of  $\mathcal{M}_1$  and  $\mathcal{M}_2$ .

### 2.2 Lagrangian Relaxation

We briefly review the *Lagrangian relaxation* approach; for a more detailed exposition, the reader is referred to [16]. The Lagrangian relaxation of the budgeted optimization problem  $\bar{II}$  is given by:

$$z(\lambda) := \text{maximize } (w(S) + \lambda(B - c(S))) \text{ subject to } S \in \mathcal{F}. \quad (\text{LR}(\lambda))$$

For any value of  $\lambda \geq 0$ , the optimal solution to  $\text{LR}(\lambda)$  gives an upper bound on the optimal solution of the original budgeted problem, because any feasible solution satisfies  $\sum_{e \in S} c(e) \leq B$ . The Lagrangian relaxation problem is to find the best such upper bound, i.e. to determine  $\lambda^*$  such that  $z(\lambda^*) = \min_{\lambda \geq 0} z(\lambda)$ . This can be done in polynomial time whenever  $\text{LR}(\lambda)$  is solvable in polynomial time [9, Theorem 24.3]. In our case, since there are combinatorial algorithms for weighted matching and weighted matroid intersection [20], we can even obtain  $\lambda^*$  in strongly polynomial time by using Megiddo’s parametric search technique [12]. Indeed, for any fixed feasible solution, the value of the Lagrangian relaxation is a linear function of  $\lambda$ , so that  $\text{LR}(\lambda)$  is the maximum of a set of linear functions, i.e. a piecewise-linear convex function. Finding the minimum of such a piecewise-linear convex function is precisely what is achieved by parametric search.

The idea behind Megiddo’s technique is that, even though we do not know  $\lambda^*$ , we can simulate the algorithm solving  $\text{LR}(\lambda^*)$  and at the same time discover  $\lambda^*$ . Towards this end, for each  $e \in E$  the value  $w_{\lambda^*}(e) := w(e) - \lambda^*c(e)$  will be manipulated symbolically as a linear function of the form  $a + \lambda^*b$ . In the simulated algorithm, which is combinatorial, these linear functions might be added together to create more linear functions, but at most a polynomial number of such functions will be used overall. Whenever the simulated algorithm asks for a comparison between some  $a + \lambda^*b$  and some  $a' + \lambda^*b'$ , we compute the critical  $\lambda$  for which  $a + \lambda b = a' + \lambda b'$ . To correctly perform the comparison and

resume the simulation of the algorithm, it is enough to know whether  $\lambda$  is smaller or larger than  $\lambda^*$ . But this can be discovered by solving one more Lagrangian subproblem, this time with weight function  $w - \lambda c$ : if the corresponding solution costs more than  $B$ , then  $\lambda < \lambda^*$ , and vice versa if the cost is larger than  $B$ . At the end of the simulation, the output of the algorithm can be used to determine  $\lambda^*$  explicitly. Finally,  $\lambda^*$  can be used to compute two solutions  $S_1, S_2$  such that:

1. Both  $S_1$  and  $S_2$  are optimal with respect to the weight function  $w_{\lambda^*}(e) := w(e) - \lambda^*c(e)$ ,  $e \in E$ ;
2.  $c(S_1) \leq B \leq c(S_2)$ .

These two solutions can be obtained by solving the relaxed problems  $LR(\lambda^* + \epsilon)$  and  $LR(\lambda^* - \epsilon)$ , respectively, for a sufficiently small  $\epsilon > 0$ . Indeed, even without knowing how small  $\epsilon$  has to be, they can be obtained by simulating again the algorithm and resolving the comparisons accordingly.

### 2.3 The Gasoline Puzzle

One crucial ingredient in our patching procedure is the solution to the puzzle cited in the introduction. We state it more formally in the following lemma.

**Lemma 3.** (*Gasoline Lemma*) *Given a sequence of  $k$  real values  $a_0, a_1, \dots, a_{k-1}$  of total value  $\sum_{j=0}^{k-1} a_j = 0$ , there is an index  $i \in \{0, 1, \dots, k - 1\}$  such that, for any  $0 \leq h \leq k - 1$ ,  $\sum_{j=i}^{i+h} a_{j \pmod k} \geq 0$ .*

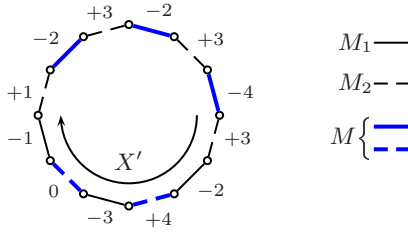
## 3 A PTAS for the Budgeted Matching Problem

In this section, we present our PTAS for the budgeted matching problem. Suppose we are given a budgeted matching instance  $I := (G, w, c, B)$ . Let  $n$  and  $m$  refer to the number of nodes and edges in  $G$ , respectively. Moreover, we define  $w_{\max} := \max_{e \in E} w(e)$  as the largest edge weight in  $I$ . Throughout this section,  $opt$  refers to the weight of an optimal solution  $M^*$  for  $I$ . In order to prove that there exists a PTAS, we proceed in two steps: First we prove that there is an algorithm to compute a feasible solution of weight at least  $opt - 2w_{\max}$ . The Gasoline Lemma will play a crucial role in this proof. The claimed PTAS is then obtained by guessing the edges of largest weight in  $M^*$  in a preliminary phase and applying the algorithm above.

**Lemma 4.** *There is a polynomial-time algorithm to compute a solution  $M$  to the budgeted matching problem of weight  $w(M) \geq opt - 2w_{\max}$ .*

*Proof.* As described in Section 2, we first compute the optimal Lagrangian multiplier  $\lambda > 0$  and two matchings  $M_1$  and  $M_2$  of maximum Lagrangian weight  $w_\lambda(M_1) = w_\lambda(M_2)$  and satisfying  $c(M_1) \leq B \leq c(M_2)$ . Observe that for  $i \in \{1, 2\}$  we have that

$$w_\lambda(M_i) + \lambda B \geq w_\lambda(M^*) + \lambda B \geq w_\lambda(M^*) + \lambda c(M^*) = opt. \tag{1}$$



**Fig. 1.** The construction used in Lemma 4. Each edge  $x_i$  is labeled with the value  $a_i$ .

We next show how to extract from  $M_1 \cup M_2$  a matching  $M$  with the desired properties in polynomial-time. Consider the symmetric difference  $M' = M_1 \oplus M_2$ . Recall that  $M' \subseteq M_1 \cup M_2$  consists of a disjoint union of paths  $\mathcal{P}$  and cycles  $\mathcal{C}$ . We apply the following procedure until eventually  $|\mathcal{P} \cup \mathcal{C}| \leq 1$ : Take some  $X \in \mathcal{P} \cup \mathcal{C}$  and let  $A := M_1 \oplus X$ . If  $c(A) \leq B$  replace  $M_1$  by  $A$ . Otherwise replace  $M_2$  by  $A$ . Observe that in each step, the cardinality of  $M_1 \cap M_2$  increases by at least one; hence this procedure terminates after at most  $O(n)$  steps. Moreover, by the optimality of  $M_1$  and  $M_2$ , the Lagrangian weight of the two matchings does not change during the process.

If at the end of this procedure  $c(M_i) = B$  for some  $i \in \{1, 2\}$ , we are done:  $M_i$  is a feasible solution to the budgeted matching problem and

$$w(M_i) = w_\lambda(M_i) + \lambda c(M_i) = w_\lambda(M_i) + \lambda B \geq \text{opt}.$$

Otherwise,  $M_1 \oplus M_2$  consists of a unique path or cycle  $X = (x_0, x_1, \dots, x_{k-1})$  such that  $c(M_1 \oplus X) = c(M_2) > B > c(M_1)$ . Consider the sequence

$$a_0 = \delta(x_0) w_\lambda(x_0), \quad a_1 = \delta(x_1) w_\lambda(x_1), \quad \dots, \quad a_{k-1} = \delta(x_{k-1}) w_\lambda(x_{k-1}),$$

where  $\delta(x_i) = 1$  if  $x_i \in M_2$  and  $\delta(x_i) = -1$  otherwise. This sequence has total value zero, because of the optimality of  $M_1$  and  $M_2$ . By the Gasoline Lemma, there must exist an edge  $x_i, i \in \{0, 1, \dots, k-1\}$ , of  $X$  such that for any cyclic subsequence  $X' = (x_i, x_{(i+1) \pmod k}, \dots, x_{(i+h) \pmod k})$ ,

$$0 \leq \sum_{j=i}^{i+h} a_{j \pmod k} = \sum_{e \in X' \cap M_2} w_\lambda(e) - \sum_{e \in X' \cap M_1} w_\lambda(e). \tag{2}$$

Consider the longest such subsequence  $X'$  satisfying  $c(M_1 \oplus X') \leq B$ . Let  $e_1 = x_i$  and  $e_2 = x_{(i+h) \pmod k}$  be the endpoints of  $X'$  (see Figure 1). Note that by the maximality of  $X'$  and by the non-negativity of the edge costs, either  $e_2 \in M_1$  or  $X'$  is a path and  $e_2$  its last edge. In both cases,  $M := (M_1 \oplus X') \setminus \{e_1\}$  is a matching (while  $M_1 \oplus X'$  might not be a matching if  $e_1 \in M_2$ ). Moreover,  $c(M) = c(M_1 \oplus X') - c(e_1) \leq c(M_1 \oplus X') \leq B$ . That is,  $M$  is a feasible solution to the budgeted matching problem.

It remains to lower bound the weight of  $M$ . We have

$$\begin{aligned} w(M_1 \oplus X') &= w_\lambda(M_1 \oplus X') + \lambda c(M_1 \oplus X') \\ &= w_\lambda(M_1 \oplus X') + \lambda B - \lambda(B - c(M_1 \oplus X')) \\ &\geq w_\lambda(M_1) + \lambda B - \lambda(B - c(M_1 \oplus X')) \\ &\geq \text{opt} - \lambda(B - c(M_1 \oplus X')), \end{aligned}$$

where the first inequality follows from (2) and the second inequality follows from (1).

Let  $e_3 = x_{(i+h+1) \pmod k}$ . The maximality of  $X'$  implies that  $c(e_3) > B - c(M_1 \oplus X') \geq 0$ . Moreover, by the optimality of  $M_1$  and  $M_2$ ,  $0 \leq w_\lambda(e_3) = w(e_3) - \lambda c(e_3)$ . Altogether  $\lambda(B - c(M_1 \oplus X')) \leq \lambda c(e_3) \leq w(e_3)$  and hence  $w(M_1 \oplus X') \geq \text{opt} - w(e_3)$ . We can thus conclude that

$$w(M) = w(M_1 \oplus X') - w(e_1) \geq \text{opt} - w(e_3) - w(e_1) \geq \text{opt} - 2w_{\max}. \quad \square$$

**Theorem 1.** *There is a PTAS for the budgeted matching problem.*

*Proof.* Let  $\epsilon \in (0, 1)$  be a given constant. Assume that the optimum matching  $M^*$  contains at least  $p := \lceil 2/\epsilon \rceil$  edges. (Otherwise the problem can be solved optimally by brute force.) Consider the following algorithm. Initially, we guess the  $p$  heaviest (with respect to weights) edges  $M_H^*$  of  $M^*$ . Then we remove from the graph  $G$  the edges in  $M_H^*$ , all edges incident to  $M_H^*$ , and all edges of weight larger than the smallest weight in  $M_H^*$ . We also decrease the budget by  $c(M_H^*)$ . Let  $I'$  be the resulting budgeted matching instance. Note that the maximum weight of an edge in  $I'$  is  $w'_{\max} \leq w(M_H^*)/p \leq \epsilon M_H^*/2$ . Moreover,  $M_L^* := M^* \setminus M_H^*$  is an optimum solution for  $I'$ . We compute a matching  $M'$  for  $I'$  using the algorithm described in the proof of Lemma 4. Eventually, we output the feasible solution  $M := M_H^* \cup M'$ .

The algorithm above has running time  $O(m^{p+O(1)}) = O(m^{O(1/\epsilon)})$ , where the  $m^p$  factor comes from the guessing of  $M_H^*$ . By Lemma 4,  $w(M') \geq w(M_L^*) - 2w'_{\max}$ . It follows that

$$\begin{aligned} w(M) &= w(M_H^*) + w(M') \geq w(M_H^*) + w(M_L^*) - 2w'_{\max} \\ &\geq w(M^*) - \epsilon w(M_H^*) \geq (1 - \epsilon)w(M^*). \end{aligned} \quad \square$$

## 4 A PTAS for the Budgeted Matroid Intersection Problem

In this section we will develop a PTAS for the budgeted matroid intersection problem. As in the PTAS for the budgeted matching problem, we will first show how to find a feasible common independent set of two matroids  $\mathcal{M}_1 = (E, \mathcal{F}_1)$  and  $\mathcal{M}_2 = (E, \mathcal{F}_2)$  of weight at least  $\text{opt} - 2w_{\max}$ , where  $w_{\max}$  is the weight of the heaviest element. The PTAS will then follow similarly as in the previous section.

Like in the matching case, we initially use Megiddo’s parametric search technique to obtain the optimal Lagrangian multiplier  $\lambda \geq 0$  and two solutions  $X, Y \in \mathcal{F}_1 \cap \mathcal{F}_2$ ,  $c(X) \leq B \leq c(Y)$ , that are optimal with respect to the Lagrangian weights  $w_\lambda(e) = w(e) - \lambda c(e)$ ,  $e \in E$ . Notice that neither  $X$  nor  $Y$  will contain any element  $e$  such that  $w_\lambda(e) < 0$ . Furthermore, both solutions can be used to derive upper bounds on the optimum solution. In fact, let  $I^*$  be the optimum solution, of weight  $opt = w(I^*)$ . For  $Z \in \{X, Y\}$ ,

$$w_\lambda(Z) + \lambda B \geq w_\lambda(I^*) + \lambda B \geq w_\lambda(I^*) + \lambda c(I^*) = opt. \tag{3}$$

If  $X$  and  $Y$  have different cardinalities, say  $|X| < |Y|$ , we extend  $\mathcal{M}_1$  and  $\mathcal{M}_2$  according to Lemma 1.4 by adding  $|Y| - |X|$  dummy elements  $D$  of weight and cost zero, and then we replace  $X$  by  $X \cup D$ . (Dummy elements will be discarded when the final solution is returned.) Of course, this does not modify the weight of the optimum solution nor the weight and cost of  $X$ . Finally, using Lemma 1.3 we truncate the two matroids to  $q := |X| = |Y|$ . The solutions  $X$  and  $Y$  will now be maximum-weight common bases of each one of the two truncated matroids.

In the following, we will show how to derive from  $X$  and  $Y$  a feasible solution of weight at least  $opt - 2w_{\max}$ . This is done in two steps. First (Section 4.1), we extract from  $X \cup Y$  two adjacent common bases, one below and the other over the budget, with the same (optimal) Lagrangian weight of  $X$  and  $Y$ . Then (Section 4.2) we apply the Gasoline Lemma to a proper auxiliary graph to compute the desired approximate solution.

### 4.1 Finding Adjacent Common Bases

The following lemma characterizes two adjacent common bases in the common basis polytope of two matroids.

**Lemma 5 (4.9).** *Assume we have two matroids  $\mathcal{M}_1 = (E, \mathcal{F}_1)$ ,  $\mathcal{M}_2 = (E, \mathcal{F}_2)$  and two common bases  $X, Y \in \mathcal{F}_1 \cap \mathcal{F}_2$ . Then  $X$  and  $Y$  are adjacent extreme points in the common basis polytope if and only if the following conditions hold:*

1. *The exchangeability graph  $\text{ex}_{\mathcal{M}_1}(X, Y)$  has a unique perfect matching  $M_1$ .*
2. *The exchangeability graph  $\text{ex}_{\mathcal{M}_2}(X, Y)$  has a unique perfect matching  $M_2$ .*
3. *The union  $M_1 \cup M_2$  forms a cycle.*

The following corollary of Lemma 5 will help us to deal with contracted matroids.

**Corollary 1.** *Let  $\mathcal{M}_1 = (E, \mathcal{F}_1)$  and  $\mathcal{M}_2 = (E, \mathcal{F}_2)$  be two matroids. Moreover, let  $Z \in \mathcal{F}_1 \cap \mathcal{F}_2$  and  $Z \subseteq X \cap Y$ . Then  $X$  and  $Y$  are adjacent extreme points in the common basis polytope of  $\mathcal{M}_1$  and  $\mathcal{M}_2$  if and only if  $X \setminus Z$  and  $Y \setminus Z$  are adjacent extreme points in the common basis polytope of  $\mathcal{M}_1/Z$  and  $\mathcal{M}_2/Z$ .*

*Proof.* First note, that  $X$  is a basis of  $\mathcal{M}_i$  if and only if  $X \setminus Z$  is a basis of  $\mathcal{M}_i/Z$  ( $i = 1, 2$ ) by Lemma 1.2. The same holds for  $Y$ . Moreover, as  $Z \subseteq X \cap Y$ , the exchangeability graphs  $\text{ex}_{\mathcal{M}_i}(X, Y)$  and  $\text{ex}_{\mathcal{M}_i/Z}(X \setminus Z, Y \setminus Z)$  ( $i = 1, 2$ ) are the same, since they are defined on the symmetric difference of  $X$  and  $Y$ . The claim then follows immediately from Lemma 5. □



Remember that  $X$  and  $Y$ , are maximum-weight common bases of  $\mathcal{M}_1$  and  $\mathcal{M}_2$  with respect to the Lagrangian weights  $w_\lambda$ , and that  $c(X) \leq B \leq c(Y)$ . Since our solution will be a subset of  $X \cup Y$ , let us delete the elements  $E' = E \setminus (X \cup Y)$  according to Lemma 4.1. In order to do a similar patching procedure as for the matching problem, we would like  $X$  and  $Y$  to be adjacent extreme points in the common basis polytope of  $\mathcal{M}_1$  and  $\mathcal{M}_2$ . The following lemma will help us to find such two adjacent common bases which are also of maximum weight with respect to  $w_\lambda$ .

**Lemma 6.** *There is a polynomial-time algorithm that finds a third maximum-weight common basis  $A$  with respect to  $w_\lambda$ , such that  $X \neq A \neq Y$  and  $X \cap Y \subseteq A \subseteq X \cup Y$ , or determines that no such basis exists.*

*Proof.* Let  $Z = X \cap Y$ . Without loss of generality, let  $X \setminus Y = \{x_1, \dots, x_r\}$  and  $Y \setminus X = \{y_1, \dots, y_r\}$ . For  $1 \leq i, j \leq r$  denote by  $\mathcal{M}_1^{ij} = \mathcal{M}_1 / Z - \{x_i, y_j\}$  and  $\mathcal{M}_2^{ij} = \mathcal{M}_2 / Z - \{x_i, y_j\}$  the matroids resulting from the contraction of  $Z$  (Lemma 4.2) and the deletion of  $x_i$  and  $y_j$  (Lemma 4.1).

Consider the following (polynomial-time) algorithm. For every  $1 \leq i, j \leq r$  compute  $A_{ij}$ , a maximum-weight common basis of  $\mathcal{M}_1^{ij}$  and  $\mathcal{M}_2^{ij}$ . If there is an  $A_{ij}$  satisfying  $|A_{ij}| = r$  and  $w_\lambda(A_{ij}) = w_\lambda(X \setminus Z)$ , then  $A = A_{ij} \cup Z$  is the desired third basis. In fact,  $A_{ij}$  is a common basis of  $\mathcal{M}_1^{ij}$  and  $\mathcal{M}_2^{ij}$ , and since  $|A| = |A_{ij}| + |Z| = |X|$ , it is also a common basis of  $\mathcal{M}_1$  and  $\mathcal{M}_2$ . Also,  $X \neq A \neq Y$  since  $x_i$  and  $y_j$  are not present in  $\mathcal{M}_1^{ij}$  and  $\mathcal{M}_2^{ij}$ .

If none of the  $A_{ij}$ 's satisfies  $|A_{ij}| = r$  and  $w_\lambda(A_{ij}) = w_\lambda(X \setminus Z)$ , then no common basis  $A$  of  $\mathcal{M}_1$  and  $\mathcal{M}_2$  with the desired properties exists. In fact, assume by contradiction that there is such a third maximum-weight basis  $A$ . Choose  $i$  and  $j$  such that  $x_i, y_j \notin A$ . Note that such indices must exist since  $X \neq A \neq Y$ . Then  $A \setminus Z$  is a common basis of  $\mathcal{M}_1^{ij}$  and  $\mathcal{M}_2^{ij}$ . Hence  $w_\lambda(A_{ij}) \geq w_\lambda(A \setminus Z)$ , since  $A_{ij}$  is a maximum-weight such common basis. Moreover  $|A_{ij}| = |A \setminus Z| = r$ , and thus  $A_{ij} \cup Z$  is a common basis of  $\mathcal{M}_1$  and  $\mathcal{M}_2$ , implying  $w_\lambda(A_{ij} \cup Z) \leq w_\lambda(A)$ . Hence  $w_\lambda(A_{ij}) \leq w_\lambda(A \setminus Z)$ . We can conclude that  $w_\lambda(A_{ij}) = w_\lambda(A \setminus Z) = w_\lambda(X \setminus Z)$ , which is a contradiction.  $\square$

We can now apply Lemma 6 as follows. Until we find a third basis  $A$ , we replace  $X$  by  $A$  if  $c(A) \leq B$ , and  $Y$  by  $A$  otherwise. In either case, the cardinality of the intersection of the new  $X$  and  $Y$  has increased. Hence this process ends in at most  $O(m)$  rounds.

At the end of the process,  $X$  and  $Y$  must be adjacent in the common basis polytope of  $\mathcal{M}_1$  and  $\mathcal{M}_2$ . In fact,  $X \setminus Y$  and  $Y \setminus X$  are maximum-weight common bases of  $\mathcal{M}_1 / (X \cap Y)$  and  $\mathcal{M}_2 / (X \cap Y)$  and there is no other maximum-weight common basis  $A'$  of  $\mathcal{M}_1 / (X \cap Y)$  and  $\mathcal{M}_2 / (X \cap Y)$ , as otherwise  $A = A' \cup (X \cap Y)$  would have been found by the algorithm from Lemma 6. Now as  $X \setminus Y$  and  $Y \setminus X$  are the only two maximum-weight common bases, they must also be adjacent on the optimal face of the common basis polytope of  $\mathcal{M}_1 / (X \cap Y)$  and  $\mathcal{M}_2 / (X \cap Y)$ . Therefore, by Corollary 1,  $X$  and  $Y$  are adjacent in the common basis polytope of  $\mathcal{M}_1$  and  $\mathcal{M}_2$ .

### 4.2 Merging Adjacent Common Bases

Let  $X$  and  $Y$  be the two adjacent solutions obtained at the end of the process described in the previous section. Notice that, if either  $c(X) = B$  or  $c(Y) = B$ , we obtain a feasible solution that is optimal also with respect to the original weights, in which case we can already stop. For this reason in the following we will assume that  $c(S_1) < B < c(S_2)$ . Without loss of generality, we also assume that  $X \setminus Y = \{x_1, x_2, \dots, x_r\}$  and  $Y \setminus X = \{y_1, y_2, \dots, y_r\}$ .

**Lemma 7.** *Given  $X$  and  $Y$  with the properties above, there is a polynomial-time algorithm which computes a common independent set  $X' \in \mathcal{F}_1 \cap \mathcal{F}_2$  such that  $c(X') \leq B$  and  $w(X') \geq \text{opt} - 2w_{\max}$ .*

*Proof.* We exploit again Lemma 5 to obtain two unique perfect matchings:  $M_1 = \{x_1y_1, \dots, x_ry_r\}$  in  $\text{ex}_{\mathcal{M}_1}(X, Y)$  and  $M_2 = \{y_1x_2, y_2x_3, \dots, y_rx_1\}$  in  $\text{ex}_{\mathcal{M}_2}(X, Y)$ . Let  $(x_1, y_1, x_2, y_2, \dots, x_r, y_r)$  be the corresponding connected cycle. Assign to the each edge  $x_jy_j$  a weight  $\delta_j := w_\lambda(y_j) - w_\lambda(x_j)$ , and weight zero to the remaining edges. Clearly  $\sum_{j=1}^r \delta_j = 0$ , since  $X$  and  $Y$  have the same maximum Lagrangian weight. Hence, by the Gasoline Lemma, there must exist an edge of the cycle such that the partial sum of the  $\delta$ -weights of each subpath starting at that edge is non-negative. Without loss of generality, assume  $x_1y_1$  is such an edge. Thus for all  $i \leq r$ ,  $\sum_{j=1}^i \delta_j \geq 0$ . Find the largest  $k \leq r$  such that

$$c(X) + \sum_{j=1}^k (c(y_j) - c(x_j)) \leq B.$$

Since  $c(Y) > B$ , we have  $k < r$  and by construction

$$c(X) + \sum_{j=1}^k (c(y_j) - c(x_j)) > B - c(y_{k+1}) + c(x_{k+1}).$$

We now show that  $X' := X \setminus \{x_1, \dots, x_{k+1}\} \cup \{y_1, \dots, y_k\}$  satisfies the claim. By the choice of  $k$ ,  $B - c_{\max} \leq B - c(y_{k+1}) < c(X') \leq B$ , where  $c_{\max} = \max_{e \in E} c(e)$ . Also, since  $\sum_{j=1}^k \delta_j \geq 0$ , we have

$$w_\lambda(X') \geq w_\lambda(X) - w_\lambda(x_{k+1}) \geq w_\lambda(X) - w_{\max}.$$

We next prove that  $X' \in \mathcal{F}_1 \cap \mathcal{F}_2$ . Consider the set  $X' \cup \{x_{k+1}\}$ : its symmetric difference with  $X$  is the set  $\{x_1, \dots, x_k\} \cup \{y_1, \dots, y_k\}$ . Recall that  $x_iy_i$  is an edge of  $M_1$ . Thus, for  $i \leq k$ , it is also an edge of  $\text{ex}_{\mathcal{M}_1}(X, X' \cup \{x_{k+1}\})$  so that this graph has a perfect matching. On the other hand this perfect matching must be unique, otherwise  $M_1$  would not be unique in  $\text{ex}_{\mathcal{M}_1}(X, Y)$ . Thus by the Exchangeability Lemma  $X' \cup \{x_{k+1}\} \in \mathcal{F}_1$ .

Similarly, consider the set  $X' \cup \{x_1\}$ : its symmetric difference with  $X$  is the set  $\{x_2, \dots, x_{k+1}\} \cup \{y_1, \dots, y_k\}$ . For  $i \leq k$ ,  $y_i x_{i+1}$  is an edge of  $M_2$ . Thus  $\text{ex}_{\mathcal{M}_2}(X, X' \cup \{x_1\})$  has a perfect matching, and it has to be unique, otherwise

$M_2$  would not be unique in  $\mathbf{ex}_{\mathcal{M}_2}(X, Y)$ . Thus by the Exchangeability Lemma  $X' \cup \{x_1\} \in \mathcal{F}_2$ . We have thus shown that  $X' \cup \{x_{k+1}\} \in \mathcal{F}_1$  and  $X' \cup \{x_1\} \in \mathcal{F}_2$ . As a consequence,  $X' \in \mathcal{F}_1 \cap \mathcal{F}_2$ .

It remains to bound the weight of  $X'$ :

$$\begin{aligned} w(X') &= w_\lambda(X') + \lambda c(X') = w_\lambda(X') + \lambda B - \lambda(B - c(X')) \\ &\geq w_\lambda(X) + \lambda B - w_{\max} - \lambda c(y_{k+1}) \geq w_\lambda(X) + \lambda B - 2w_{\max} \\ &\geq \mathit{opt} - 2w_{\max}. \end{aligned}$$

Above we used the fact that  $w_\lambda(e) \geq 0$  for all  $e \in Y$ , so in particular  $w_\lambda(y_{k+1}) = w(y_{k+1}) - \lambda c(y_{k+1}) \geq 0$ , implying  $w_{\max} \geq w(y_{k+1}) \geq \lambda c(y_{k+1})$ . The last inequality follows from (3). □

**Theorem 2.** *The budgeted matroid intersection problem admits a PTAS.*

*Proof.* Let  $\epsilon \in (0, 1)$  be a given constant. Assume that the optimum solution contains at least  $p := \lceil 2/\epsilon \rceil$  elements (otherwise the problem can be solved optimally by brute force). We first guess the  $p$  elements of largest weight in the optimal solution. Using contraction (Lemma 112) we remove these elements from both matroids, and using deletion (Lemma 111) we as well remove all elements that have a larger weight than any of the contracted elements. We decrease the budget by an amount equal to the cost of the guessed elements. Finally we apply the above algorithm and we add back the guessed elements to the solution. The final solution will have weight at least  $\mathit{opt} - 2w'_{\max}$ , where  $w'_{\max}$  is the largest weight of the elements that remained after the guessing step. Since  $\mathit{opt} \geq (2/\epsilon)w'_{\max}$ , we obtain a solution of weight at least  $(1 - \epsilon)\mathit{opt}$ . The running time of the algorithm above is  $O(m^{O(1/\epsilon)})$ . □

## 5 Concluding Remarks and Open Problems

There are several problems that we left open. One natural question is whether we can apply our patching technique to other budgeted problems. Apparently, the main property that we need is that the difference between two solutions that are adjacent in the solution polytope of the corresponding unbudgeted problem can be characterized by a proper alternating path or cycle. This deserves further investigation.

Another natural question is whether there are fully polynomial-time approximation schemes for the problems considered here. We conjecture that budgeted matching is not strongly NP-hard. However, finding an FPTAS for that problem might be a very difficult task. In fact, for polynomial weights and costs, the budgeted matching problem is equivalent to the *exact perfect matching problem* (proof is given in the appendix): Given an undirected graph  $G = (V, E)$ , edge weights  $w : E \rightarrow \mathbb{Q}$ , and a parameter  $W \in \mathbb{Q}$ , find a perfect matching of weight exactly  $W$ , if any. This problem was first posed by Papadimitriou and Yannakakis [17]. For polynomial weights, the problem admits a polynomial-time Monte Carlo algorithm [13, 3]. Hence, it is very unlikely that exact perfect

matching with polynomial weights is NP-hard (since this would imply  $RP=NP$ ). However, after 25 years, the problem of finding a deterministic algorithm to solve this problem is still open.

Finally, an interesting open problem is whether our approach can be extended to the case of multiple budget constraints. The difficulty here is that the Gasoline Lemma alone seems not able to fill in the cost-budget-gap for several budget constraints at the same time.

*Acknowledgements.* The authors would like to thank Jochen Könemann, Rajiv Raman, and Martin Skutella for helpful discussions.

## References

1. Aggarwal, V., Aneja, Y.P., Nair, K.P.K.: Minimal spanning tree subject to a side constraint. *Computers & Operations Research* 9(4), 287–296 (1982)
2. Beasley, J., Christofides, N.: An algorithm for the resource constrained shortest path problem. *Networks* 19, 379–394 (1989)
3. Camerini, P., Galbiati, G., Maffioli, F.: Random pseudo-polynomial algorithms for exact matroid problems. *Journal of Algorithms* 13, 258–273 (1992)
4. Frank, A., Tardos, É.: Generalized polymatroids and submodular flows. *Mathematical Programming* 42, 489–563 (1988)
5. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York (1979)
6. Guignard, M., Rosenwein, M.B.: An application of Lagrangean decomposition to the resource-constrained minimum weighted arborescence problem. *Networks* 20(3), 345–359 (1990)
7. Hassin, R., Levin, A.: An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem using matroid intersection. *SIAM Journal on Computing* 33(2), 261–268 (2004)
8. Hong, S.-P., Chung, S.-J., Park, B.H.: A fully polynomial bicriteria approximation scheme for the constrained spanning tree problem. *Operations Research Letters* 32(3), 233–239 (2004)
9. Iwata, S.: On matroid intersection adjacency. *Discrete Mathematics* 242, 277–281 (2002)
10. Lovász, L.: *Combinatorial Problems and Exercises*. North-Holland, Amsterdam (1979)
11. Marathe, M.V., Ravi, R., Sundaram, R., Ravi, S.S., Rosenkrantz, D.J., Hunt III, H.B.: Bicriteria network design problems. In: *Proc. 22nd Int. Colloquium on Automata, Languages and Programming*, pp. 487–498 (1995)
12. Megiddo, N.: Combinatorial optimization with rational objective functions. *Mathematics of Operations Research* 4(4), 414–424 (1979)
13. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. *Combinatorica* 7(1), 105–113 (1987)
14. Naor, J., Shachnai, H., Tamir, T.: Personal communication (2007)
15. Naor, J., Shachnai, H., Tamir, T.: Real-time scheduling with a budget. *Algorithmica* 47(3), 343–364 (2007)
16. Nemhauser, G.L., Wolsey, L.A.: *Integer and Combinatorial Optimization*. John Wiley, Chichester (1988)

17. Papadimitriou, C., Yannakakis, M.: The complexity of restricted spanning tree problems. *Journal of the ACM* 29(2), 285–309 (1982)
18. Ravi, R., Goemans, M.X.: The constrained minimum spanning tree problem (extended abstract). In: *Proc. 5th Scandinavian Workshop on Algorithms and Theory*, pp. 66–75 (1996)
19. Schrijver, A.: *Theory of Linear and Integer Programming*. Wiley, Chichester (1986)
20. Schrijver, A.: *Combinatorial Optimization. Polyhedra and Efficiency*. Springer, Heidelberg (2003)
21. Shmoys, D.B., Tardos, É.: Scheduling unrelated machines with costs. In: *Proc. 4th Symposium on Discrete Algorithms*, pp. 448–454 (1993)

## A Exact Perfect Matching and Budgeted Matching

We prove that the budgeted matching problem and the exact perfect matching problem are equivalent for polynomial weights and costs. We call the variant of the budgeted matching problem, where we additionally require that the computed matching is perfect, the *budgeted perfect matching problem*.

**Lemma 8.** *For polynomial weights and costs, the following problems are polynomially reducible: (a) exact perfect matching; (b) budgeted perfect matching; (c) budgeted matching.*

*Proof.* Without loss of generality, we assume that all weights and costs are non-negative integers. Let  $w_{\max}$  and  $c_{\max}$  be the largest weight and cost, respectively.

(a)  $\Rightarrow$  (b): Let  $(G, w, W)$  be an exact perfect matching instance. Solve the budgeted perfect matching instance  $(G, w, c, B)$ , where  $B = W$  and  $c(e) = w(e)$  for every edge  $e$ . If the solution returned has weight smaller than  $B = W$ , the original problem is infeasible. Otherwise, the solution computed is a perfect matching of  $G$  of weight  $W$ .

(c)  $\Rightarrow$  (a): Let  $(G, w, c, B)$  be a budgeted matching instance. For two given  $W^*$  and  $B^*$ , consider the exact perfect matching instance  $(G, w', W')$ , where  $W' = (n/2 + 1)c_{\max}W^* + B^*$  and  $w'(e) = (n/2 + 1)c_{\max}w(e) + c(e)$  for every edge  $e$ . The problem  $(G, w', W')$  is feasible if and only if there is a matching of weight  $W^*$  and cost  $B^*$  in the original problem. By trying all the (polynomially many) possible values for  $W^*$  and  $B^*$ , we obtain the desired solution to the original problem.

(b)  $\Rightarrow$  (c): Let  $(G, w, c, B)$  be a budgeted perfect matching instance. Consider the budgeted matching instance  $(G, w', c, B)$ , where  $w'(e) = w(e) + (n/2 + 1)w_{\max}$  for every edge  $e$ . The original problem is feasible if and only if the maximum matching  $M^*$  of the new problem contains  $n/2$  edges, i.e.,  $M^*$  is a perfect matching.  $\square$

# Primal-Dual Schema for Capacitated Covering Problems\*

Tim Carnes and David Shmoys

Cornell University, Ithaca NY 14853, USA  
tcarnes@orie.cornell.edu, shmoys@cs.cornell.edu

**Abstract.** Primal-dual algorithms have played an integral role in recent developments in approximation algorithms, and yet there has been little work on these algorithms in the context of LP relaxations that have been strengthened by the addition of more sophisticated valid inequalities. We introduce primal-dual schema based on the LP relaxations devised by Carr, Fleischer, Leung & Phillips for the minimum knapsack problem as well as for the single-demand capacitated facility location problem. Our primal-dual algorithms achieve the same performance guarantees as the LP-rounding algorithms of Carr et al., which rely on applying the ellipsoid algorithm to an exponentially-sized LP. Furthermore, we introduce new flow-cover inequalities to strengthen the LP relaxation of the more general capacitated single-item lot-sizing problem; using just these inequalities as the LP relaxation, we obtain a primal-dual algorithm that achieves a performance guarantee of 2.

## 1 Introduction

Primal-dual algorithms have played an integral role in recent developments in approximation algorithms, and yet there has been little work on these algorithms in the context of LP relaxations that have been strengthened by the addition of more sophisticated valid inequalities. We introduce primal-dual schema based on the LP relaxations devised by Carr, Fleischer, Leung & Phillips [5] for the minimum knapsack problem as well as for the single-demand capacitated facility location problem. Our primal-dual algorithms achieve the same performance guarantees as the LP-rounding algorithms of Carr et al., which rely on applying the ellipsoid algorithm to an exponentially-sized LP. Furthermore, we introduce new flow-cover inequalities to strengthen the LP relaxation of the more general capacitated single-item lot-sizing problem; using just these inequalities as the LP relaxation, we obtain a primal-dual algorithm that achieves a performance guarantee of 2.

We say an algorithm is an approximation algorithm with a performance guarantee of  $\alpha$  when the algorithm runs in polynomial time and always produces a solution with cost within a factor of  $\alpha$  of optimal. Primal-dual algorithms are able

---

\* Research supported partially by NSF grants CCR-0635121, CCR-0430682 & DMI-0500263.

to gain the benefits of LP-based techniques, such as automatically generating a new lower bound for each problem instance, but without having to solve an LP. Primal-dual approximation algorithms were developed by Bar-Yehuda & Even and Chvátal for the weighted vertex cover and set cover problems, respectively [3,7]. Subsequently, this approach has been applied to many other combinatorial problems, such as results of Agrawal, Klein & Ravi [2], Goemans & Williamson [10], Bertsimas & Teo [4] and Levi, Roundy & Shmoys [12] for other related covering problems. Other recent work has been done on covering problems with capacity constraints by Even, Levi, Rawitz, Schieber, Shahar & Sviridenko [8], Chuzhoy & Naor [6] and Gandhi, Halperin, Khuller, Kortsarz & Srinivasan [9].

In developing primal-dual (or any LP-based) approximation algorithms, it is important to have a strong LP formulation for the problem. However, there are cases when the LP relaxation of the natural IP formulation for a problem has a large integrality gap. When this happens, one can mend the situation by introducing extra *valid* inequalities that hold for any solution to the problem, but restrict the feasible region of the LP. One class of valid inequalities that has proved useful for a variety of problem are called *flow-cover inequalities*. A large class of flow-cover inequalities was developed by Aardal, Pochet & Wolsey [1] for the capacitated facility location problem. Carr et al. [5] developed a different style of flow-cover inequalities for simpler capacitated covering problems which we use in developing our primal-dual algorithms. Levi, Lodi & Sviridenko [11] used a subset of the flow-cover inequalities of Aardal et al. [1] to develop a LP-rounding algorithm for the multiple-item lot-sizing problem with monotone holding costs. Our model is not a special case of theirs, however, since our result allows time-dependent order capacities whereas their result assumes constant order capacities across all periods. Following this work and the development of our own flow-cover inequalities for lot-sizing problems, Sharma & Williamson [13] demonstrated that the result of Levi et al. [11] has an analogue based on our flow-cover inequalities as well. Finally, it is worth noting that Van Hoesel & Wagelmans [14] have a FPTAS for the single-item lot-sizing problem that makes use of dynamic programming and input data rounding. The disadvantage of this result is that the running time of the algorithm can be quite slow for particular performance guarantees. Although the primal-dual algorithm we develop is a 2-approximation algorithm, this is a worst-case bound and we would expect it to perform much better on average in practice. Also this is the first LP-based result for the case of time-dependent capacities.

The three models studied in this paper are generalizations of one another. That is to say, the minimum knapsack problem is a special case of the single-demand capacitated facility location problem, which is a special case of the single-item lot-sizing problem. We present the result in order of generality with the aim of explaining our approach in the simplest setting first. The *minimum knapsack problem* gives a set of items, each with a weight and a value. The objective is to find a minimum-weight subset of items such that the total value of the items in the subset meets some specified demand. In the *single-demand*

*facility location problem* there is a set of facilities, each with an opening cost and a capacity, as well as a per-unit serving cost that must be paid for each unit of demand a facility serves. The goal is to open facilities to completely serve a specified amount of demand, while minimizing the total service and facility opening costs. Finally the *single-item lot-sizing problem* considers a finite planning period of consecutive time periods. In each time period there is a specified level of demand, as well as a potential order with a given capacity and order opening cost. A feasible solution must open enough orders and order enough inventory so that in each time period there is enough inventory to satisfy the demand of that period. The inventory is simply the inventory of the previous time period plus however much is ordered in the current time period, if any. However, a per-unit holding cost is incurred for each unit of inventory held over a given time period.

The straightforward LP relaxations for these problems have a bad integrality gap, but can be strengthened by introducing valid flow-cover inequalities. The inequalities Carr et al. [5] developed for the minimum knapsack problem are as follows

$$\sum_{i \in F \setminus A} u_i(A) y_i \geq D - u(A) \quad \forall A \subseteq F,$$

where the  $y_i$  are the binary decision variables indicating if item  $i$  is chosen,  $u(A)$  is the total value of the subset of items  $A$ , and the  $u_i(A)$  can be thought of as the effective value of item  $i$  with respect to  $A$ , which is the minimum of the actual value and the right-hand-side of the inequality. These inequalities arise by considering that if we did choose all items in the set  $A$ , then we still have an induced subproblem on all of the remaining items, and the values can be truncated since we are only concerned with integer solutions. Our primal-dual algorithm works essentially as a modified greedy algorithm, where at each stage the item is selected that has the largest value per cost. Instead of the actual values and costs, however, we use the effective values and the slacks of the dual constraints as costs. Similar to the greedy algorithm for the traditional maximum-value knapsack problem, the last item selected and everything selected beforehand, can each be bounded in cost by the dual LP value, yielding a 2-approximation.

The remainder of this paper is organized as follows. In Section 2 we go over the minimum knapsack result in more detail. In Section 3 we generalize this result to apply to the single-demand capacitated facility location problem. Finally in Section 4 we generalize the flow-cover inequalities to handle the lot-sizing problem, and then present and analyze a primal-dual algorithm for the single-item lot-sizing problem.

## 2 Minimum Knapsack

In the *minimum knapsack problem* one is given a set of items  $F$ , and each item  $i \in F$  has a value  $u_i$  and a weight  $f_i$ . The goal is to select a minimum weight



subset of items,  $S \subseteq F$ , such that the value of  $S$ ,  $u(S)$ , is at least as big as a specified demand,  $D$ . The natural IP formulation for this problem is

$$\begin{aligned} \text{opt}_{MK} &:= \min \sum_{i \in F} f_i y_i && \text{(MK-IP)} \\ \text{s.t.} & \sum_{i \in F} u_i y_i \geq D && (1) \\ & y_i \in \{0, 1\} && \forall i \in F, \end{aligned}$$

where the  $y_i$  variables indicate if item  $i$  is chosen. The following example from [5] demonstrates that the integrality gap between this IP and the LP relaxation is at least as bad as  $D$ . Consider just 2 items where  $u_1 = D - 1$ ,  $f_1 = 0$ ,  $u_2 = D$  and  $f_2 = 1$ . The only feasible integer solution chooses both items and has a cost of 1, whereas the LP solution can set  $y_1 = 1$  and  $y_2 = 1/D$  and incurs a cost of only  $1/D$ . To remedy this situation we consider using the flow-cover inequalities introduced in [5].

The idea is to consider a subset of items  $A \subseteq F$  such that  $u(A) < D$ , and let  $D(A) = D - u(A)$ . This means that even if all of the items in the set  $A$  are chosen, we must choose enough items in  $F \setminus A$  such that the demand  $D(A)$  is met. This is just another minimum knapsack problem where the items are restricted to  $F \setminus A$  and the demand is now  $D(A)$ . The value of every item can be restricted to be no greater than the demand without changing the set of feasible integer solutions, so let  $u_i(A) = \min\{u_i, D(A)\}$ . This motivates the following LP

$$\begin{aligned} \text{opt}_{MKP} &:= \min \sum_{i \in F} f_i y_i && \text{(MK-P)} \\ \text{s.t.} & \sum_{i \in F \setminus A} u_i(A) y_i \geq D(A) && \forall A \subseteq F \\ & y_i \geq 0 && \forall i \in F, \end{aligned} \tag{2}$$

and by the validity of the flow-cover inequalities argued above we have that every feasible integer solution to (MK-IP) is a feasible solution to (MK-P). The dual of this LP is

$$\begin{aligned} \text{opt}_{MKD} &:= \max \sum_{A \subseteq F} D(A) v(A) && \text{(MK-D)} \\ \text{s.t.} & \sum_{A \subseteq F: i \notin A} u_i(A) v(A) \leq f_i && \forall i \in F \\ & v(A) \geq 0 && \forall A \subseteq F. \end{aligned} \tag{3}$$

Our primal-dual algorithm begins by initializing all of the primal and dual variables to zero, which produces a feasible dual solution and an infeasible primal integer solution. Taking our initial subset of items,  $A$ , to be the empty set, we

increase the dual variable  $v(A)$ . Once a dual constraint becomes tight, the item corresponding to that constraint is added to the set  $A$ , and we now increase the new variable  $v(A)$ . Note that increasing  $v(A)$  does not increase the left-hand-sides of dual constraints corresponding to items in  $A$ , so dual feasibility will not be violated. This process is repeated as long as  $D(A) > 0$ , and once we finish we call our final set of items  $S$ , which is our integer solution. This is a feasible solution to (MK-IP) since  $D(S) \leq 0$ , which implies  $u(S) \geq D$ .

---

**Algorithm 1:** Primal-Dual for Minimum Knapsack

---

```

 $y, v \leftarrow 0$ 
 $A \leftarrow \emptyset$ 
while  $D(A) > 0$  do
    Increase  $v(A)$  until a dual constraint becomes tight for item  $i$ 
     $y_i \leftarrow 1$ 
     $A \leftarrow A \cup \{i\}$ 
 $S \leftarrow A$ 

```

---

**Theorem 1.** Algorithm 1 terminates with a solution of cost no greater than  $2 \cdot \text{opt}_{MK}$ .

*Proof.* Let  $\ell$  denote the final item selected by Algorithm 1. Then because the algorithm only continues running as long as  $D(A) > 0$  we have that

$$D(S \setminus \{\ell\}) > 0 \Rightarrow D - u(S \setminus \{\ell\}) > 0 \Rightarrow u(S \setminus \{\ell\}) < D.$$

Also, the variable  $v(A)$  is positive only if  $A \subseteq S \setminus \{\ell\}$ . Thus, since the algorithm only selects items for which the constraint (3) has become tight, we have that the cost of the solution produced is

$$\sum_{i \in F} f_i y_i = \sum_{i \in S} f_i = \sum_{i \in S} \sum_{A \subseteq F: i \notin A} u_i(A) v(A).$$

The expression on the right-hand side is summing over all items,  $i$ , in the final solution  $S$ , and all subsets of items,  $A$ , that do not contain item  $i$ . This is the same as summing over all subsets of items,  $A$ , and all items in the final solution that are not in  $A$ . Thus we can reverse the order of the summations to obtain

$$\begin{aligned} \sum_{i \in F} f_i y_i &= \sum_{A \subseteq F} v(A) \sum_{i \in S \setminus A} u_i(A) \\ &= \sum_{A \subseteq F} v(A) (u(S \setminus \{\ell\}) - u(A) + u_\ell(A)) \\ &< \sum_{A \subseteq F} v(A) (D - u(A) + u_\ell(A)), \end{aligned}$$

where the inequality follows by making use of our observation above that  $u(S \setminus \{\ell\}) < D$ . But we also have  $u_\ell(A) \leq D(A)$  by definition, hence

$$\sum_{i \in F} f_i y_i \leq \sum_{A \subseteq F} 2D(A)v(A) \leq 2 \cdot \text{opt}_{MKD}. \quad \square$$

### 3 Single-Demand Facility Location

In the *single-demand facility location problem*, one is given a set of facilities  $F$ , where each facility  $i \in F$  has capacity  $u_i$ , opening cost  $f_i$ , and there is a per-unit cost  $c_i$  to serve the demand, which requires  $D$  units of the commodity. The goal is to select a subset of facilities to open,  $S \subseteq F$ , such that the combined cost of opening the facilities and serving the demand is minimized. The natural IP formulation for this problem is

$$\text{opt}_{FL} := \min \sum_{i \in F} (f_i y_i + Dc_i x_i) \tag{FL-IP}$$

$$\text{s.t. } \sum_{i \in F} x_i = 1 \tag{4}$$

$$u_i y_i \geq D x_i \tag{5}$$

$$y_i \geq x_i \quad \forall i \in F \tag{6}$$

$$y_i \in \{0, 1\} \quad \forall i \in F$$

$$x_i \geq 0 \quad \forall i \in F,$$

where each  $y_i$  indicates if facility  $i \in F$  is open and each  $x_i$  indicates the fraction of  $D$  being served by facility  $i \in F$ . The same example from the minimum knapsack problem also demonstrates the large integrality gap of this IP. We once again turn to the flow-cover inequalities introduced by Carr et al. [5].

For these inequalities, we once again consider a subset of facilities  $A \subseteq F$  such that  $u(A) < D$ , and let  $D(A) = D - u(A)$ . This means that even if all of the facilities in the set  $A$  are opened, we must open enough facilities in  $F \setminus A$  such that we will be able to assign the remaining demand  $D(A)$ . But certainly for any feasible integer solution, a facility  $i \in F \setminus A$  cannot contribute more than  $\min\{Dx_i, u_i(A)y_i\}$  towards the demand  $D(A)$ . So if we partition the remaining orders of  $F \setminus A$  into two sets  $F_1$  and  $F_2$ , then for each  $i \in F_1$  we will consider its contribution as  $Dx_i$ , and for each  $i \in F_2$  we will consider its contribution as  $u_i(A)y_i$ . The total contribution of these facilities must be at least  $D(A)$ , so if we let  $\mathcal{F}$  be the set of all 3-tuples that partition  $F$  into three sets, we obtain the following LP

$$\text{opt}_{FLP} := \min \sum_{i \in F} (f_i y_i + Dc_i x_i) \tag{FL-P}$$

$$\text{s.t. } \sum_{i \in F_1} Dx_i + \sum_{i \in F_2} u_i(A)y_i \geq D(A) \quad \forall (F_1, F_2, A) \in \mathcal{F} \tag{7}$$

$$x_i, y_i \geq 0 \quad \forall i \in F,$$

and by the validity of the flow-cover inequalities argued above we have that every feasible integer solution to (FL-IP) is a feasible solution to (FL-P). The dual of this LP is

$$\text{opt}_{FLD} := \max \sum_{(F_1, F_2, A) \in \mathcal{F}} D(A)v(F_1, F_2, A) \tag{FL-D}$$

$$\text{s.t.} \quad \sum_{(F_1, F_2, A) \in \mathcal{F}: i \in F_1} Dv(F_1, F_2, A) \leq Dc_i \quad \forall i \in F \tag{8}$$

$$\sum_{(F_1, F_2, A) \in \mathcal{F}: i \in F_2} u_i(A)v(F_1, F_2, A) \leq f_i \quad \forall i \in F \tag{9}$$

$$v(F_1, F_2, A) \geq 0 \quad \forall (F_1, F_2, A) \in \mathcal{F}.$$

As in Section 2 the primal-dual algorithm begins with all variables at zero and an empty subset of facilities,  $A$ . Before a facility is added to  $A$ , we will require that it become tight on both types of dual constraints. To achieve this we will leave each facility in  $F_1$  until it becomes tight on constraint (8), move it into  $F_2$  until it is also tight on constraint (9), and only then move it into  $A$ . As before the algorithm terminates once the set  $A$  has enough capacity to satisfy the demand, at which point we label our final solution  $S$ .

---

**Algorithm 2:** Primal-Dual for Single-Demand Facility Location

---

```

x, y, v ← 0
F1 ← F
F2, A ← ∅
while D(A) > 0 do
    Increase v(F1, F2, A) until a dual constraint becomes tight for facility i
    if i ∈ F1 then /* i tight on (8) but not on (9) */
        | Move i from F1 into F2
    else /* else i tight on (8) and (9) */
        | xi ← ui(A)/D
        | yi ← 1
        | Move i from F2 into A
S ← A

```

---

Clearly Algorithm 2 terminates with a feasible solution to (FL-IP) since all of the demand is assigned to facilities that are fully opened.

**Theorem 2.** *Algorithm 2 terminates with a solution of cost no greater than  $2 \cdot \text{opt}_{FL}$ .*

*Proof.* Let  $\ell$  denote the final facility selected by Algorithm 2. By the same reasoning as in Section 2 we have

$$D(S \setminus \{\ell\}) > 0 \Rightarrow D - u(S \setminus \{\ell\}) > 0 \Rightarrow u(S \setminus \{\ell\}) < D.$$

The variable  $v(F_1, A)$  is positive only if  $A \subseteq S \setminus \{\ell\}$ . If a facility is in  $S$  then it must be tight on both constraints (8) and (9) so

$$\begin{aligned} \sum_{i \in F} (f_i y_i + D c_i x_i) &= \sum_{i \in S} (f_i + D c_i x_i) \\ &= \sum_{i \in S} \left[ \sum_{(F_1, F_2, A) \in \mathcal{F}: i \in F_2} u_i(A) v(F_1, F_2, A) + x_i \sum_{(F_1, F_2, A) \in \mathcal{F}: i \in F_1} D v(F_1, F_2, A) \right], \end{aligned}$$

as in Section 2 we can simply reverse the order of summation to get

$$\sum_{i \in F} (f_i y_i + D c_i x_i) = \sum_{(F_1, F_2, A) \in \mathcal{F}} v(F_1, F_2, A) \left[ \sum_{i \in S \cap F_2} u_i(A) + \sum_{i \in S \cap F_1} D x_i \right].$$

Recall that at the last step of Algorithm 2, facility  $\ell$  was assigned  $D(S \setminus \{\ell\})$  amount of demand. Since  $D(A)$  only gets smaller as the algorithm progresses, we have that regardless of what summation above the facility  $\ell$  is in, it contributes no more than  $D(A)$ . All of the other terms can be upper bounded by the actual capacities and hence

$$\begin{aligned} \sum_{i \in F} (f_i y_i + D c_i x_i) &= \sum_{(F_1, F_2, A) \in \mathcal{F}} v(F_1, F_2, A) [u(S \setminus \{\ell\}) - u(A) + D(A)] \\ &< \sum_{(F_1, F_2, A) \in \mathcal{F}} 2D(A) v(F_1, F_2, A) \leq 2 \cdot \text{opt}_{FLD}, \end{aligned}$$

where the strict inequality above follows from the observation made earlier.  $\square$

### 4 Single-Item Lot-Sizing with Linear Holding Costs

In the single-item lot-sizing problem, one is given a planning period consisting of time periods  $F := \{1, \dots, T\}$ . For each time period  $t \in F$ , there is a demand,  $d_t$ , and a potential order with capacity  $u_t$ , which costs  $f_t$  to place, regardless of the amount of product ordered. At each period, the total amount of product left over from the previous period plus the amount of product ordered during this period must be enough to satisfy the demand of this period. Any remaining product is held over to the next period, but incurs a cost of  $h_t$  per unit of product stored. If we let

$$h_{st} = \sum_{r=s}^{t-1} h_r$$

and set  $h_{tt} = 0$  for all  $t \in F$ , then we obtain a standard IP formulation for this problem as follows

$$\text{opt}_{LS} := \min \sum_{s=1}^T f_s y_s + \sum_{s=1}^T \sum_{t=s}^T h_{st} d_t x_{st} \tag{LS-IP}$$

$$\text{s.t. } \sum_{s=1}^t x_{st} = 1 \quad \forall t \tag{10}$$

$$\sum_{t=s}^T d_t x_{st} \leq u_s y_s \quad \forall s \tag{11}$$

$$x_{st} \leq y_s \quad \forall s \leq t \tag{12}$$

$$y_s \in \{0, 1\} \quad \forall s$$

$$x_{st} \geq 0 \quad \forall s \leq t.$$

where the  $y_s$  variables indicate if an order has been placed at time period  $s$ , and the  $x_{st}$  variables indicate what fraction of the demand  $d_t$  is being satisfied from product ordered during time period  $s$ . This formulation once again suffers from a bad integrality gap, which can be demonstrated by the same example as in the previous two sections. We introduce new flow-cover inequalities to strengthen this formulation.

The basic idea is similar to the inequalities used in sections 2 and 3. We would like to consider a subset of orders,  $A$ , where even if we place all the orders in  $A$  and use these orders to their full potential, there is still unmet demand. In the previous cases, the amount of unmet demand was  $D(A) = D - u(A)$ . Now, however, that is not quite true, since each order  $s$  is capable of serving only the demand points  $t$  where  $t \geq s$ . Instead, we now also consider a subset of demand points  $B$ , and define  $d(A, B)$  to be the total unmet demand in  $B$ , when the orders in  $A$  serve as much of the demand in  $B$  as possible. More formally

$$d(A, B) := \min d(B) - \sum_{s \in A} \sum_{t \geq s: t \in B} d_t x_{st} \tag{RHS-LP}$$

$$\text{s.t. } \sum_{s=1}^t x_{st} \leq 1 \quad \forall t \tag{13}$$

$$\sum_{t=s}^T d_t x_{st} \leq u_s \quad \forall s \tag{14}$$

$$x_{st} \geq 0 \quad \forall s \leq t.$$

As before, we would also like to restrict the capacities of the orders not in  $A$ . To do this, we define

$$u_s(A, B) := d(A, B) - d(A \cup \{s\}, B), \tag{15}$$

which is the decrease in remaining demand that would result if order  $s$  were added to  $A$ . (This reduces to the same  $u_s(A)$  as defined in the previous sections when considered in the framework of the earlier problems.) We once again partition the remaining orders in  $F \setminus A$  into two sets,  $F_1$  and  $F_2$ , and count the

contribution of orders in  $F_1$  as  $\sum_t d_t x_{st}$  and orders in  $F_2$  as  $u_s(A, B)y_s$ . This leads to the following LP, where once again  $\mathcal{F}$  is the set of all 3-tuples that partition  $F$  into three sets.

$$\begin{aligned}
 \text{opt}_{LSP} := \min & \sum_{s=1}^T f_s y_s + \sum_{s=1}^T \sum_{t=s}^T h_{st} d_t x_{st} & (\text{LS-P}) \\
 \text{s.t.} & \sum_{\substack{s \in F_1, \\ t \in B}} d_t x_{st} + \sum_{s \in F_2} u_s(A, B)y_s \geq d(A, B) & (16) \\
 & \forall (F_1, F_2, A) \in \mathcal{F}, B \subseteq F \\
 & x_{st}, y_s \geq 0 & \forall s, t.
 \end{aligned}$$

**Lemma 1.** Any feasible solution to (LS-IP) is a feasible solution to (LS-P).

*Proof.* Consider a feasible integer solution  $(x, y)$  to (LS-IP) and let  $S := \{s : y_s = 1\}$ . Now for any  $(F_1, F_2, A) \in \mathcal{F}$  and  $B \subseteq F$  we know

$$\sum_{\substack{s \in F_1, \\ t \in B}} d_t x_{st} \geq d((F_2 \cap S) \cup A, B),$$

since there is no way to assign demand from  $B$  to orders in  $(F_2 \cap S) \cup A$  without leaving at least  $d((F_2 \cap S) \cup A, B)$  amount of demand unfulfilled. Thus at least that amount of demand must be served by the other orders in  $S$ , namely those in  $F_1$ . Let  $k := |F_2 \cap S|$  and let  $s_1, \dots, s_k$  denote the elements of that set in some order. Furthermore let  $S_i := \{s_1, \dots, s_i\}$  for each  $1 \leq i \leq k$ , so  $S_k = F_2 \cap S$ . Then by repeated use of (15) we have

$$\begin{aligned}
 \sum_{\substack{s \in F_1, \\ t \in B}} d_t x_{st} & \geq d((F_2 \cap S) \cup A, B) \\
 & = d(((F_2 \cap S) \cup A) \setminus S_1, B) - u_{s_1}(((F_2 \cap S) \cup A) \setminus S_1, B) \\
 & = d(((F_2 \cap S) \cup A) \setminus S_k, B) - \sum_{i=1}^k u_{s_i}(((F_2 \cap S) \cup A) \setminus S_i, B) \\
 & = d(A, B) - \sum_{i=1}^k u_{s_i}(((F_2 \cap S) \cup A) \setminus S_i, B) \\
 & \geq d(A, B) - \sum_{s \in F_2 \cap S} u_s(A, B) \\
 & \geq d(A, B) - \sum_{s \in F_2} u_s(A, B)y_s,
 \end{aligned}$$

where the inequalities follow since  $u_s(A, B)$  is increasing as elements are removed from  $A$ . Thus  $(x, y)$  satisfies all of the flow-cover inequalities (16).  $\square$

The dual of (LS-P) is

$$\text{opt}_{LSD} := \max \sum_{(F_1, F_2, A) \in \mathcal{F}} d(A, B)v(F_1, F_2, A, B) \tag{LS-D}$$

$$\text{s.t.} \sum_{\substack{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F: \\ s \in F_1, t \in B}} v(F_1, F_2, A, B) \leq h_{st} \quad \forall s \leq t \tag{17}$$

$$\sum_{\substack{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F: \\ s \in F_2}} u_s(A, B)v(F_1, F_2, A, B) \leq f_s \quad \forall s \tag{18}$$

$$v(F_1, F_2, A, B) \geq 0 \quad \forall (F_1, F_2, A) \in \mathcal{F}, B \subseteq F,$$

where we simply divided constraint (17) by  $d_t$ .

Before we get to a primal-dual algorithm, we must first introduce some notation and associated machinery.

$$e_t := d_t \left[ 1 - \sum_{r=1}^t x_{rt} \right] \text{ - amount of demand currently unsatisfied in period } t$$

We define  $\text{Fill}(A, B)$  to be the following procedure that describes how to assign demand from  $B$  to orders in  $A$ . We consider the orders in  $A$  in arbitrary order, and for each order we serve as much demand as possible, processing demands from earliest to latest.

In the previous two sections, there was effectively only one demand, and so we never had to be concerned about how demand is assigned once an item or facility is chosen. Now there are many demand points, and so we must be careful that as we maintain our solution set  $A$ , we are serving as much demand as possible from the orders in  $A$ . The way the primal-dual algorithm will assign demand will correspond with how the  $\text{Fill}$  procedure works, and we show that this is a maximal assignment.

**Lemma 2.** *If we start from an empty demand assignment and run  $\text{Fill}(A, B)$ , then we obtain a demand assignment such that  $e(B) = d(A, B)$ . Thus,  $\text{Fill}$  produces an assignment that is optimal for (RHS-LP).*

*Proof.* Consider the latest time period  $t \in B$  where  $e(t) > 0$ . All orders at time periods at or before  $t$  must be serving up to capacity, since otherwise they could have served more of demand  $d_t$ . All orders after time period  $t$  could not have served any more demand in  $B$  since all demand points in  $B$  after  $t$  are fully served. □

Just as in the previous two sections, the primal-dual algorithm initializes the variables to zero and the set  $A$  to the empty set. As in Section 3, we initialize  $F_1$  to be the set of all orders, and an order will become tight first on constraint (17), when it will be moved to  $F_2$ , and then tight on (18), when it will be moved to  $A$ . Unlike in Section 3, however, constraint (17) consists of many different inequalities for the same order. This difficulty is averted since all the constraints (17)



for a particular order will become tight at the same time, as is proved below in Lemma 3. This is achieved by slowly introducing demand points into the set  $B$ . Initially,  $B$  will consist only of the last demand point,  $T$ . Every time an order becomes tight on all constraints (17), it is moved from  $F_1$  into  $F_2$ , and the demand point of that time period is added to  $B$ . In this way we always maintain that  $F_1$  is a prefix of  $F$ , and  $B$  is the complementary suffix. When an order  $s$  becomes tight on constraint (18), we move it to  $A$  and assign demand to it by running the procedure  $\text{Fill}(s, B)$ . Additionally we create a *reserve set* of orders,  $R_s$ , for order  $s$ , that consists of all orders earlier than  $s$  that are not in  $F_1$  at the time  $s$  was added to  $A$ . Finally, once all of the demand has been assigned to orders, we label the set of orders in  $A$  as our current solution,  $S^*$ , and now enter a clean-up phase. We consider the orders in the reverse order in which they were added to  $A$ , and for each order,  $s$ , we check to see if there is enough remaining capacity of the orders that are in both the reserve set and our current solution,  $S^* \cap R_s$ , to take on the demand being served by  $s$ . If there is, then we reassign that demand to the orders in  $S^* \cap R_s$  arbitrarily and remove  $s$  from our solution  $S^*$ . When the clean-up phase is finished we label the nodes in  $S^*$  as our final solution,  $S$ .

---

**Algorithm 3:** Primal-Dual for Single-Item Lot-Sizing

---

```

 $x_{st}, y_s \leftarrow 0$ 
 $F_1 \leftarrow F$ 
 $F_2, A \leftarrow \emptyset$ 
 $B \leftarrow \{T\}$ 
while  $d(A, F) > 0$  do
    Increase  $v(F_1, F_2, A, B)$  until dual constraint becomes tight for order  $s$ 
    if  $s \in F_1$  then                                     /*  $s$  tight on (17) but not on (18) */
        Move  $s$  from  $F_1$  into  $F_2$ 
         $B \leftarrow F \setminus F_1$ 
    else                                                 /* else  $s$  tight on (17) and (18) */
         $y_s \leftarrow 1$ 
         $\text{Fill}(s, B)$ 
        Move  $s$  from  $F_2$  into  $A$ 
         $R_s \leftarrow \{r \in F \setminus F_1 : r < s\}$ 
 $S^* \leftarrow A$                                          /* start clean-up phase */
for  $s \leftarrow$  last order added to  $A$  to first order added to  $A$  do
    if remaining capacity of orders in  $S^* \cap R_s$  is enough to serve demand of  $s$ 
    then
        Remove  $s$  from solution  $S^*$ 
         $y_s, x_{st} \leftarrow 0$                                /* unassign demand of  $s$  */
         $\text{Fill}(S^* \cap R_s, F)$                              /* reassign to reserve orders */
 $S \leftarrow S^*$ 

```

---

**Lemma 3.** All of the constraints (17) for a particular order become tight at the same time, during the execution of Algorithm 3.

*Proof.* We instead prove an equivalent statement: when demand  $t$  is added to  $B$ , then for any order  $s \leq t$  the slack of the constraint (17) corresponding to  $s$  and demand  $t'$  is  $h_{st}$  for any  $t' \geq t$ . This statement implies the lemma by considering  $s = t$ , which implies all constraints (17) for  $s$  become tight at the same time. We prove the above statement by (backwards) induction on the demand points. The case where  $t = T$  clearly holds, since this demand point is in  $B$  before any dual variable is increased, and hence the slack of constraint (17) for order  $s$  and demand  $T$  is  $h_{sT}$ . Now assume the statement holds for some  $t \leq T$ . If we consider order  $s = t - 1$  then by the inductive hypothesis the slack of all constraints (17) for  $s$  and demand  $t' \geq t$  is  $h_{st}$ . Hence the slack of all constraints (17) for orders  $s' \leq s$  decreases by  $h_{st}$  between the time  $t$  is added to  $B$  and when  $t - 1$  is added to  $B$ . Then by the inductive hypothesis again, we have that for any order  $s' \leq s$  and any demand  $t' \geq t$ , when  $t - 1$  is added to  $B$  the slack of the corresponding constraint (17) is

$$h_{s't} - h_{st} = \sum_{r=s'}^{t-1} h_r - \sum_{r=s}^{t-1} h_r = \sum_{r=s'}^{t-1} h_r - h_{t-1} = \sum_{r=s'}^{t-2} h_r = h_{s',t-1}.$$

Hence the statement also holds for  $t - 1$ . □

Define  $\ell$  to be

$$\ell = \ell(F_1, F_2, A) := \max\{s : s \in S \cap F_2\} \cup \{0\},$$

so  $\ell$  is the latest order in the final solution that is also in  $F_2$ , for a given partition, or if there is no such order then  $\ell$  is 0, which is a dummy order with no capacity.

**Lemma 4.** *Upon completion of Algorithm 3, for any  $F_1, F_2, A, B$  such that  $v(F_1, F_2, A, B) > 0$ , we have*

$$\sum_{s \in S \cap F_2} u_s(A, B) + \sum_{s \in S \cap F_1} \sum_{t \in B: t \geq s} d_t x_{st} < d(A, B) + u_\ell(A, B).$$

*Proof.* First we consider the case when  $S \cap F_2 \neq \emptyset$ . Here the first summation is empty, so we just need to show the bound holds for the second summation. We know that any order  $s \in S \cap F_1$  is not in the reserve set for any order in  $A$ . This follows since  $F_1$  decreases throughout the course of the algorithm, so since  $s$  is in  $F_1$  at this point, then clearly  $s$  was in  $F_1$  at any point previously, in particular when any order in  $A$  was originally added to  $A$ . Hence no demand that was originally assigned to an order in  $A$  was ever reassigned to an order in  $F_1$ . But from Lemma 2 we know that only an amount  $d(A, B)$  of demand from demand points in  $B$  is not assigned to orders in  $A$ , thus orders in  $F_1$  can serve at most this amount of demand from demand points in  $B$  and we are done.

Otherwise it must be the case that  $S \cap F_2 \neq \emptyset$ , hence  $\ell$  corresponds to a real order that was not deleted in the clean-up phase. By the way  $\ell$  was chosen we know that any other order in  $S \cap F_2$  is in an earlier time period than  $\ell$ , and since these orders were moved out of  $F_1$  before  $\ell$  was added to  $A$ , they must

be in the reserve set  $R_\ell$ . However, since  $\ell$  is in the final solution, it must be the case that when  $\ell$  was being considered for deletion the orders in the reserve set that were still in the solution did not have sufficient capacity to take on the demand assigned to  $\ell$ . Thus if we let  $x'$  denote the demand assignment during the clean-up phase when  $\ell$  was being considered, then

$$\sum_{s \in (S \cap F_2) \setminus \{\ell\}} u_s(A, B) \leq u((S \cap F_2) \setminus \{\ell\}) < \sum_{\substack{s \in S \cap F_2 \\ t \in B: t \geq s}} d_t x'_{st}.$$

None of the orders in  $A$  had been deleted when  $\ell$  was being considered for deletion, so only an amount  $d(A, B)$  of the demand in  $B$  was being served by orders outside of  $A$ . As argued previously, none of the orders in  $F_1$  took on demand being served by orders in  $A$ , but they also did not take on demand being served by orders in  $S \cap F_2$ , since these orders were never deleted. Thus we can upper bound the amount of demand that orders from  $S \cap F_2$  could have been serving at the time order  $\ell$  was being considered for deletion as follows

$$\sum_{\substack{s \in S \cap F_2 \\ t \in B: t \geq s}} d_t x'_{st} \leq d(A, B) - \sum_{\substack{s \in S \cap F_1 \\ t \in B: t \geq s}} d_t x_{st}.$$

The desired inequality is obtained by rearranging terms and adding  $u_\ell(A, B)$  to both sides. □

We are now ready to analyze the cost of the solution.

**Theorem 3.** *Algorithm 3 terminates with a solution of cost no greater than  $2 \cdot \text{opt}_{LS}$ .*

*Proof.* As in the previous two sections, we can use the fact that all of the orders in the solution are tight on all constraints (17) and (18).

$$\begin{aligned} & \sum_{s \in S} \left[ f_s + \sum_{t=s}^T d_t h_{st} x_{st} \right] \\ &= \sum_{s \in S} \left[ \sum_{\substack{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F: \\ s \in F_2}} u_s(A, B) v(F_1, F_2, A, B) \right. \\ & \qquad \qquad \qquad \left. + \sum_{t=s}^T d_t x_{st} \sum_{\substack{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F: \\ s \in F_1, t \in B}} v(F_1, F_2, A, B) \right] \\ &= \sum_{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F} v(F_1, F_2, A, B) \left[ \sum_{s \in S \cap F_2} u_s(A, B) + \sum_{s \in S \cap F_1} \sum_{t \in B: t \geq s} d_t x_{st} \right]. \end{aligned}$$

Now we can apply Lemma 4 and achieve the desired result:

$$\begin{aligned} \sum_{s \in S} \left[ f_s + \sum_{t=s}^T d_t h_{st} x_{st} \right] &< \sum_{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F} v(F_1, F_2, A, B) [d(A, B) + u_\ell(A, B)] \\ &\leq \sum_{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F} 2d(A, B)v(F_1, F_2, A, B) \\ &\leq 2 \cdot \text{opt}_{LSD}. \quad \square \end{aligned}$$

**Acknowledgement.** We would like to thank Retsef Levi for many helpful discussions.

## References

1. Aardal, K., Pochet, Y., Wolsey, L.A.: Capacitated facility location: valid inequalities and facets. *Math. Oper. Res.* 20(3), 562–582 (1995)
2. Agrawal, A., Klein, P., Ravi, R.: When trees collide: an approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Comput.* 24(3), 440–456 (1995)
3. Bar-Yehuda, R., Even, S.: A linear-time approximation algorithm for the weighted vertex cover problem. *J. Algorithms* 2(2), 198–203 (1981)
4. Bertsimas, D., Teo, C.P.: From valid inequalities to heuristics: a unified view of primal-dual approximation algorithms in covering problems. *Operations Research* 46(4), 503–514 (2003)
5. Carr, R.D., Fleischer, L., Leung, V.J., Phillips, C.A.: Strengthening integrality gaps for capacitated network design and covering problems. In: *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 106–115 (2000)
6. Chuzhoy, J., Naor, J.: Covering problems with hard capacities. *SIAM J. Comput.* 36(2), 498–515 (2006)
7. Chvátal, V.: A greedy heuristic for the set-covering problem. *Math. Oper. Res.* 4(3), 233–235 (1979)
8. Even, G., Levi, R., Rawitz, D., Schieber, B., Shahar, S., Sviridenko, M.: Algorithms for capacitated rectangle stabbing and lot sizing with joint set-up costs. *ACM Trans. on Algorithms* (submitted, 2007)
9. Gandhi, R., Halperin, E., Khuller, S., Kortsarz, G., Srinivasan, A.: An improved approximation algorithm for vertex cover with hard capacities. *Journal of Computer and System Sciences* 72(1), 16–33 (2006)
10. Goemans, M.X., Williamson, D.P.: A general approximation technique for constrained forest problems. *SIAM J. Comput.* 24(2), 296–317 (1995)
11. Levi, R., Lodi, A., Sviridenko, M.: Approximation algorithms for the multi-item capacitated lot-sizing problem via flow-cover inequalities. In: Fischetti, M., Williamson, D.P. (eds.) *IPCO 2007*. LNCS, vol. 4513, pp. 454–468. Springer, Heidelberg (2007)
12. Levi, R., Roundy, R., Shmoys, D.B.: Primal-dual algorithms for deterministic inventory problems. *Math. Oper. Res.* 31(2), 267–284 (2006)
13. Sharma, Y., Williamson, D.: Personal communication (2007)
14. van Hoesel, C.P.M., Wagelmans, A.P.M.: Fully polynomial approximation schemes for single-item capacitated economic lot-sizing problems. *Math. Oper. Res.* 26(2), 339–357 (2001)

# Offline and Online Facility Leasing<sup>\*</sup>

Chandrashekhara Nagarajan and David P. Williamson

School of Operations Research and Information Engineering, Cornell University,  
Ithaca, NY 14853, USA

chandra@orie.cornell.edu, dpw@cs.cornell.edu

**Abstract.** We study the problem of leasing facilities over time, following the general infrastructure leasing problem framework introduced by Anthony and Gupta [1]. If there are  $K$  different lease types, Anthony and Gupta give an  $O(K)$ -approximation algorithm for the problem. We are able to improve this to a 3-approximation algorithm by using a variant of the primal-dual facility location algorithm of Jain and Vazirani [5]. We also consider the online version of the facility leasing problem, in which the clients to be served arrive over time and are not known in advance. This problem generalizes both the online facility location problem (introduced by Meyerson [6]) and the parking permit problem (also introduced by Meyerson [7]). We give a deterministic algorithm for the problem that is  $O(K \log n)$ -competitive. To achieve our result, we modify an  $O(\log n)$ -competitive algorithm of Fotakis [2] for the online facility location problem.

## 1 Introduction

A current trend in business and logistics is that of a non-asset owning company: for example, a trucking company that owns no trucks, but instead subcontracts the work out to other trucking companies, and makes its profits by efficiently combining the various trucking jobs that it receives [8]. Such organizations rely heavily on optimization to find the efficiencies that make their companies profitable. Some of the optimization problems that arise in such settings have been well-studied in the literature, but some lead to previously unconsidered variations. For instance, in the location theory literature, it is usually assumed that the facilities opened are available to serve customer demand from the moment of opening onwards. However, a non-asset owning company may instead prefer to serve customers by subcontracting or leasing the facilities needed to serve demands at some point in time, then allowing the lease to elapse as demand subsides.

In this paper, we will consider a formalization of this problem known as the *facility leasing problem*, a variant of the well-studied uncapacitated facility location problem; this variant was introduced by Anthony and Gupta [1]. In this problem, we are given a set  $F$  of potential facilities we may lease, and a set  $D$  of potential clients. For each potential client  $j \in D$  and potential facility  $i \in F$ ,

---

<sup>\*</sup> Both authors supported in part by NSF CCF-0514628.

there is a cost  $c_{ij}$  associated with serving client  $j$  from facility  $i$ . We assume these costs obey the triangle inequality, which is a reasonable assumption if the clients and facilities are in a metric space, and the service cost is dependent on the distance between the client and facility. There are distinct time periods, and in each time period  $t$ , a set  $D_t \subseteq D$  of clients arrives and must be served from some facility open during time period  $t$ . There are  $K$  different possible lease lengths,  $l_1, \dots, l_K$ , and we can lease a facility  $i \in F$  starting at any time period  $t$  for lease length  $l_k$  at a cost  $f_i^k$ ; that is, the facility  $i$  is then open during time units in  $[t, t + l_k)$  and can serve any client  $j$  arriving during this time interval at cost  $c_{ij}$ . Furthermore, the cost of the lease is allowed to depend both on the facility and the lease type. The goal is to minimize the cost of the leases plus the total cost of serving each client in each set  $D_t$  from some facility open during time  $t$ .

In the *offline* version of the problem, we are given the number of time periods  $T$  as input, as well as the set of clients  $D_t$  for each time period. In the *online* version of the problem, we do not know the number of time periods, nor do we know the client sets. In each new time period  $t$ , we see the client set  $D_t$ , and we can sign new leases starting at time  $t$  or some time prior (e.g. we can sign a lease for the month of November in mid-November). We then must assign each client in  $D_t$  to some facility open at time  $t$ .

Since the offline version of the problem contains the uncapacitated facility location problem as a special case, the problem is NP-hard. We therefore study *approximation algorithms* for the problem. An  $\alpha$ -approximation algorithm runs in polynomial time and finds a solution of cost at most  $\alpha$  times the value of an optimal solution. For the online version of the problem, we use the notion of *competitive ratio* to evaluate our algorithms. We say that an algorithm is  $\alpha$ -competitive if its cost after  $T$  time steps is no more than  $\alpha$  times the value of an optimal solution to the offline problem.

The offline facility leasing problem was introduced by Anthony and Gupta [1] in the context of more general *infrastructure leasing* problems. They show an interesting and surprising connection of infrastructure leasing problems with  $K$  lease types to  $K$ -stage stochastic optimization problems. In particular, given an  $\alpha$ -approximation algorithm for the  $K$ -stage stochastic problem, they derive an  $O(\alpha)$ -approximation algorithm for the related leasing problem. For the facility leasing problem, therefore, they obtain an  $O(K)$ -approximation algorithm from an  $O(K)$ -approximation algorithm for the  $K$ -stage stochastic facility location problem due to Swamy and Shmoys [9].

Our first result is a 3-approximation algorithm for the offline facility leasing problem. To achieve this, we use a modification of the primal-dual algorithm for the uncapacitated facility location problem due to Jain and Vazirani [5]. It is relatively straightforward to give a linear programming relaxation of the facility leasing problem; for the dual of this LP, we increase dual variables uniformly for each client/time pair  $(j, t)$  for  $j \in D_t$  until a dual constraint associated with a facility lease becomes tight. As with Jain and Vazirani, the trick to obtaining a good approximation algorithm is to open a subset of the tight leases. In our

case, however, the aspect of time makes this step of the algorithm slightly more complicated. We resolve this by signing longer leases than the LP indicates; we are then able to obtain the same guarantee as Jain and Vazirani.

We then turn to the online version of the problem. The online facility leasing problem generalizes two problems introduced by Meyerson, *the online facility location problem* [6] and *the parking permit problem* [7]. In *the online facility location problem*, once a facility is opened, it is open for the rest of time; we can consider this as a facility leasing problem with a single lease type, where the lease length is indefinite. Meyerson [6] gives an  $O(\log n)$ -competitive algorithm for the problem, where  $n$  is the number of clients that appear. This is improved by Fotakis [3] to an  $O(\frac{\log n}{\log \log n})$ -competitive algorithm; furthermore, the author shows that no better competitive ratio is possible for the problem. Another paper of Fotakis [2] gives a simple  $O(\log n)$ -competitive algorithm for the problem whose choices are guided by solutions to dual of a linear programming relaxation of the problem. Curiously, the competitive ratio of the algorithm is not analyzed in terms of this dual. In the *parking permit problem*, a professor walks to work on days when it is sunny, and drives when it rains. If she drives, then she must have a parking permit valid for that day. Parking permits can be purchased for various lengths of time, and given that the weather is unknown, the question is which permits to buy and when. This problem corresponds to the online facility leasing with a single facility  $i$ , a single client  $j$ , and a service cost  $c_{ij} = 0$ ; buying a parking permit corresponds to leasing the facility  $i$  for the corresponding length of time and a rainy day corresponds to a demand of client  $j$  in that time period. Meyerson gives a deterministic  $O(K)$ -competitive algorithm for the parking permit problem, and a randomized  $O(\log K)$ -competitive algorithm. He also shows that any deterministic algorithm has competitive ratio  $\Omega(K)$ , and any randomized algorithm has competitive ratio  $\Omega(\log K)$ .

Our second main result is an  $O(K \log n)$ -competitive algorithm for the online facility leasing problem. No previous algorithm is known for this problem. We first reanalyze the simple  $O(\log n)$ -competitive algorithm of Fotakis by using a *dual fitting* argument. We show that his algorithm can be viewed as constructing an infeasible solution to the dual of a linear relaxation of the facility leasing problem, such that the cost of the primal solution given by the online algorithm is at most a constant factor times the value of the dual. We then show that scaling the dual variables down by  $O(\log n)$  causes it to become feasible, so that the primal cost is at most  $O(\log n)$  times the value of a feasible dual, giving the result. We then give a modification of Fotakis' algorithm to the case of facility leasing. Our algorithm is a generalization of both Fotakis' and Meyerson's algorithms in the sense that for the online facility location problem it reduces to Fotakis' algorithm and for the parking permit problem it reduces to Meyerson's algorithm.

Our paper is structured as follows. In Section 2, we introduce the linear programming relaxation we will use throughout the paper, as well as some basic definitions and terminology. In Section 3, we give our offline 3-approximation algorithm for the problem. Section 4 gives Fotakis'  $O(\log n)$ -competitive algorithm

for the online facility location algorithm and our analysis of it. Section 5 gives our extension of Fotakis’ algorithm to the online facility leasing problem. We give some open problems in Section 6. Some proofs are omitted for space reasons.

## 2 Definitions and Terminology

We have a set of potential facilities locations  $F$  which can be opened to serve any subset of clients  $D$  which arrive over a period of time from 1 to  $T$ . In the offline problem  $T$  is part of the input, while for the online problem  $T$  is unknown. For every time period  $t \in [T]$  every client in the set  $D_t$  wants to be served by a facility that is open at time  $t$ . A facility can be leased for an interval  $[t, t + l_k)$  at a cost of  $f_i^k$  for  $k \in \mathcal{L}$  where  $\mathcal{L}$  is the different set of leases available and any client can be served by the facility leased during this particular interval. For simplicity of notation, we let  $I_t^k$  denote the time interval  $[t, t + l_k)$ . We let  $K = |\mathcal{L}|$ . We abuse the definition of a facility and say that triple  $(i, k, t)$  is a facility at location  $i \in F$  leased for a duration of  $l_k$  starting at time  $t$ . Similarly, a client  $(j, t)$  represents  $j \in D_t$  seeking a facility from which to be served at time  $t$ . For simplicity, we’ll denote the set of facility triples  $(i, k, t)$  as  $\mathcal{F}$ , and the set of client demand pairs  $(j, t)$  as  $\mathcal{D}$ . We introduce here some notation we will use throughout the paper. We use  $(a)_+ \equiv \max(0, a)$ . For a set  $X \subseteq F$  and client  $j \in D$ , we define  $c(X, j) = \min_{i \in X} c_{ij}$ . For a set  $X \subseteq \mathcal{F}$  and client  $(j, t) \in \mathcal{D}$ , we define  $c(X, (j, t)) = \min_{(i, k, t') \in X, t \in I_{t'}^k} c_{ij}$ .

In the offline version of the problem, we seek to find a set of facilities (the facility locations and their leasing intervals) so as to minimize the sum of opening costs of the facilities and the cost of connecting the clients to the nearest facility open at that time. So we seek to find a set  $\mathcal{T} \subseteq \mathcal{F}$  of facilities so as to minimize  $\sum_{(i, k, t) \in \mathcal{T}} f_i^k + \sum_{(j, t) \in \mathcal{D}} \min_{(i, k, t') \in \mathcal{T}, t \in I_{t'}^k} c_{ij}$ .

In the online version of the problem, we must irrevocably assign each client in  $D_t$  to some facility open at time  $t$  before we see the clients in  $D_{t+1}$ . If we later open another, closer facility to  $(j, t)$  than the one it was originally assigned to, we are not allowed to change its assignment.

In both cases, the following linear program is a relaxation of the problem. The variable  $y_{ikt}$  indicates whether we open the facility  $i$  with lease type  $k$  at time  $t$ . The variable  $x_{ikt',jt}$  indicates whether client  $(j, t)$  is assigned to facility  $(i, k, t')$  to be served.

$$\text{Min} \quad \sum_{(i, k, t) \in \mathcal{F}} f_i^k y_{ikt} + \sum_{(j, t) \in \mathcal{D}} \sum_{(i, k, t') \in \mathcal{F}: t \in I_{t'}^k} c_{ij} x_{ikt',jt}$$

subject to:

$$\begin{aligned} x_{ikt',jt} &\leq y_{ikt} && \forall (i, k, t') \in \mathcal{F}, (j, t) \in \mathcal{D} \\ \sum_{(i, k, t') \in \mathcal{F}: t \in I_{t'}^k} x_{ikt',jt} &\geq 1 && \forall (j, t) \in \mathcal{D} \\ x_{ikt',jt}, y_{ikt} &\geq 0 && \forall (j, t) \in \mathcal{D}, (i, k, t), (i, k, t') \in \mathcal{F}. \end{aligned}$$



Taking the dual, we have

$$\begin{aligned}
 & \text{Max} \quad \sum_{(j,t) \in \mathcal{D}} v_{jt} \\
 & \text{subject to:} \\
 & \quad \sum_{(j,t) \in \mathcal{D}} w_{ikt',jt} \leq f_i^k \quad \forall (i, k, t') \in \mathcal{F} \tag{1} \\
 & \quad v_{jt} - w_{ikt',jt} \leq c_{ij} \quad \forall (j, t) \in \mathcal{D}, (i, k, t') \in \mathcal{F}, t \in I_{t'}^k \\
 & \quad v_{jt}, w_{ikt',jt} \geq 0 \quad \forall (j, t) \in \mathcal{D}, (i, k, t') \in \mathcal{F}.
 \end{aligned}$$

Our algorithms will work by using the dual to construct an integer solution to the primal problem. In the offline case, we construct a feasible dual such that the primal costs no more than 3 times the value of the dual. In the online case, we will construct an infeasible dual solution such that the primal costs no more than  $K + 1$  times the value of the dual, and scaling the dual variables  $v_{jt}$  down by a factor of  $O(\log n)$  makes the dual solution feasible. This gives us our two central results.

### 3 An Algorithm for Offline Facility Leasing

In this section, we give a 3-approximation algorithm for the offline facility leasing problem. The approach we follow is similar to the primal-dual facility location algorithm by Jain and Vazirani [5]. The dual LP used is given in Section 2.

#### 3.1 The Algorithm

As in [5], the algorithm proceeds in two phases. In Phase 1, the algorithm operates in a primal-dual fashion determining a set of temporarily open facilities (triplets) and assigns each client  $(j, t)$  to a temporarily open facility. In Phase 2 the algorithm chooses a subset of these facilities to open permanently and reassigns the clients to the permanently open facilities. Phase 1 of our algorithm is simply the extension of [5] to our setting; the main difference of our algorithm is the set of (facility, lease type, time) triplets chosen to open in Phase 2.

Following [5], in the first phase we uniformly increase the dual variables  $v_{jt}$  associated with the clients  $(j, t) \in \mathcal{D}$ . We implicitly maintain the dual variables  $w_{ikt',jt} = (v_{jt} - c_{ij})_+$ . At some point, we will no longer be able to increase the dual variables and maintain dual feasibility, for one of two reasons. First, some constraint (1) will become *tight* for some facility  $(i, k, t) \in \mathcal{F}$ ; that is, the dual constraint is met with equality. In this case we declare the triple  $(i, k, t)$  to be temporarily open. Let  $\mathcal{T}$  be the set of temporarily open facilities. Second, we might have  $v_{jt} = c_{ij}$  for some temporarily open facility  $(i, k, t')$  with  $t \in I_{t'}^k$ ; we can't increase  $v_{jt}$  further since this would force  $w_{ikt',jt} > 0$  and violate the corresponding constraint (1). We say that a client  $(j, t)$  *contributes to* facility  $(i, k, t')$  if  $t \in I_{t'}^k$  and  $v_{jt} > c_{ij}$ ; it is *connected to* the facility if  $t \in I_{t'}^k$

and  $v_{jt} \geq c_{ij}$ . After either event happens, we continue increasing the duals of all clients not connected to a temporarily open facility. Eventually, all clients are connected to a temporarily open facility, and Phase 1 ends.

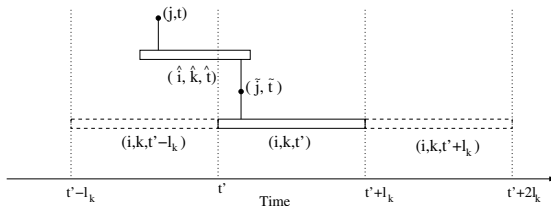
In Phase 1, a client might have contributed towards opening multiple facilities. However we want to ensure that a client contributes only to a single facility lease. Phase 2 ensures this by opening only a subset of these temporarily open facilities. To run Phase 2, we construct a graph  $G(V, E)$  with vertex set  $V$  as the set of temporarily opened facilities in  $\mathcal{T}$  in Phase 1. We add an edge between two facilities in  $G$  if there is a client that contributes to both the facilities. As in [5], we now find a maximal independent set in  $G$ ; here, however, we give priority for facilities with longer lease length to be in the independent set. In other words, we order the temporarily open facilities according to non-increasing lease lengths then greedily add facilities to the independent set following this order. This gives an independent set  $\mathcal{I} \subseteq \mathcal{T}$  with the following properties:

1. The independent set is maximal.
2. For every temporarily opened facility not in the independent set there is a facility in the independent set with same or longer lease length adjacent to it in the graph  $G$ .

Given the set  $\mathcal{I}$ , for each triple  $(i, k, t) \in \mathcal{I}$ , we sign three leases, the one corresponding to  $(i, k, t)$ , then the two leases at facility  $i$  of type  $k$  that start at time  $t + l_k$  and end at time  $t$ ; that is, we open  $(i, k, t)$ ,  $(i, k, t + l_k)$ , and  $(i, k, (t - l_k)_+)$ . Let the set of facilities opened be  $\mathcal{I}'$ . Note that  $|\mathcal{I}'| = 3 \cdot |\mathcal{I}|$ .

### 3.2 The Analysis

Consider any client  $(j, t)$ . If it is connected to a facility  $(i, k, t') \in \mathcal{I}$  then we assign  $(j, t)$  to that facility and say that  $(j, t)$  is *directly* connected to  $(i, k, t')$ . If it is not connected to a facility in  $\mathcal{I}$ , then since  $(j, t)$  connected to some temporarily open facility  $(\hat{i}, \hat{k}, \hat{t})$ , and  $\mathcal{I}$  is a maximal independent set in  $G$ , this temporarily open facility must be adjacent to some  $(i, k, t') \in \mathcal{I}$  via an edge of  $G$ . We will *indirectly* connect client  $(j, t)$  to one of the three facilities opened corresponding to  $(i, k, t')$ . We claim that  $t \in [(t' - l_k)_+, t' + l_k)$ , so that one of these facilities can serve  $(j, t)$ . See Fig. 1 for an illustration. To prove the claim, observe that



**Fig. 1.** Illustration:  $(j, t)$  indirectly connected to  $(i, k, t')$ ,  $t \in [(t' - l_k)_+, t' + 2l_k)$

since there is an edge between  $(\hat{i}, \hat{k}, \hat{t})$  and  $(i, k, t')$  in  $G$ , there is some client  $(\tilde{j}, \tilde{t})$  that contributes to both facilities. This implies that  $\tilde{t} \in I_{\hat{i}}^{\hat{k}} \cap I_{t'}^k$ . Furthermore, by property (2) of  $\mathcal{I}$ , the length of the lease  $l_{\hat{k}}$  of the temporarily opened facility  $(\hat{i}, \hat{k}, \hat{t})$  is no longer than the lease  $l_k$ . Thus the interval  $I_{\hat{i}}^{\hat{k}} \subseteq [(t' - l_k)_+, t' + l_k)$ . Since  $(j, t)$  connects to  $(\hat{i}, \hat{k}, \hat{t})$ , then  $t \in I_{\hat{i}}^{\hat{k}} \subseteq [(t' - l_k)_+, t' + l_k)$ , and our claim is proven.

We show that three times the sum of dual variables of the clients pays for the facility leasing costs and the cost of serving clients from the nearest open facility. For each client  $(j, t)$  directly connected to a facility  $(i, k, t')$ , let  $v_{jt}^f = v_{jt} - c_{ij}$  and let  $v_{jt}^s = c_{ij}$ . For each client  $(j, t)$  indirectly connected to a facility  $(i, k, t')$ , let  $v_{jt}^f = 0$  and  $v_{jt}^s = v_{jt}$ . Note that  $v_{jt} = v_{jt}^f + v_{jt}^s$  for all clients  $(j, t)$ .

**Lemma 1.** *For each facility  $(i, k, t')$  in  $\mathcal{I}$ ,*

$$\sum_{(j,t) \text{ directly connected to } (i,k,t')} v_{jt}^f = f_i^k.$$

**Lemma 2**

$$\sum_{(i,k,t) \in \mathcal{I}'} f_i^k = 3 \cdot \sum_{(j,t) \in \mathcal{D}} v_{jt}^f.$$

**Lemma 3.** *For every client  $(j, t)$  indirectly connected to a facility  $(i, k, t')$ , the connection cost  $c_{ij} \leq 3 \cdot v_{jt}^s$ .*

Let  $y_{ikt} = 1$  for each  $(i, k, t) \in \mathcal{I}'$  and  $y_{ikt} = 0$  otherwise. For each client  $(j, t)$  let  $(i, k, t')$  be the facility in  $\mathcal{I}'$  which it is connected to, directly or indirectly. Set  $x_{ikt',jt} = 1$  for all such client-facility pair. Note that  $(\mathbf{x}, \mathbf{y})$  is primal feasible. The theorem below follows directly from the preceding discussion.

**Theorem 1.** *The primal feasible solution  $(x, y)$  and the dual feasible solution  $(v, w)$  satisfy*

$$\sum_{(i,k,t) \in \mathcal{F}} f_i^k y_{ikt} + \sum_{(j,t) \in \mathcal{D}} \sum_{(i,k,t) \in \mathcal{F}: t \in I_{t'}^k} c_{ij} x_{ikt',jt} \leq 3 \cdot \sum_{(j,t) \in \mathcal{D}} v_{jt} \leq 3 \cdot OPT.$$

*Thus the algorithm is a 3-approximation algorithm for the offline facility leasing problem.*

## 4 Fotakis' Online Facility Location Algorithm

In this section, we give Fotakis' algorithm for the online facility location problem, and restate the analysis of it as a dual-fitting argument. The algorithm will construct an infeasible dual solution to the dual of Section 2 such that the cost of the primal solution constructed is at most twice the dual objective. We'll then show that scaling the dual variables down by a factor of  $O(\log n)$  will give a feasible dual solution, yielding the competitive ratio of the algorithm. In the next section, we'll show how a modification of Fotakis' algorithm and this analysis gives our result for the online facility leasing problem.

### 4.1 The Algorithm

Note that in the case of the online facility location problem, we have a single facility lease type and its duration is infinite; once a facility is opened it continues to remain open. We denote the cost of the facility at  $i$  by  $f_i$ .

The algorithm works as follows. We maintain an (infeasible) set of dual variables  $v_{jt}$ , one for each client  $j \in D_t$  for all time periods thus far. We also maintain a set  $X$  of facilities that have been opened so far, which is initially empty. Each client  $(j, t)$  that has arrived at some prior point in time and been connected to some facility in  $X$  bids  $(c(X, j) - c_{ij})_+$  towards the facility cost of each unopened facility  $i$ . When a set of clients  $D_t$  arrives, we sequence through the clients  $j \in D_t$  in index order. We increase the dual variable  $v_{jt}$ ; client  $(j, t)$  bids  $(v_{jt} - c_{ij})_+$  towards the cost of each unopened facility  $i$ . We continue increasing  $v_{jt}$  until either  $v_{jt} = c_{i'j}$  for some previously opened facility  $i' \in X$ , or the sum of the bids on some unopened facility  $i$  is equal to its facility cost  $f_i$ . In the former case, we assign client  $(j, t)$  to the facility  $i'$ ; in the latter case, we open facility  $i$ , add  $i$  to  $X$ , and assign  $(j, t)$  to  $i$ . We then continue to the next facility in  $D_t$ . Note that client  $(j, t)$  immediately reduces its bids on other unopened facilities  $\hat{i}$  from  $(v_{jt} - c_{i\hat{j}})_+$  to  $(c(X, j) - c_{i\hat{j}})_+ = (c_{ij} - c_{i\hat{j}})_+$ . The algorithm is summarized in Algorithm 1. The algorithm is very similar to an algorithm of Jain et al. [4] for the (offline) uncapacitated facility location problem, except that in that algorithm all clients increase their dual variables simultaneously (since all are known in advance), and clients whose bids are used to open an unopened facility are then reassigned to that facility.

---

**Algorithm 1.** Fotakis' algorithm for online facility location

---

```

 $X \leftarrow \emptyset; D \leftarrow \emptyset; t \leftarrow 0$ 
While true
  For each  $j \in D_t$ 
    Increase  $v_{jt}$  until  $v_{jt} = c_{ij}$  for  $i \in X$  or  $(v_{jt} - c_{ij})_+ + \sum_{(j', t') \in D} (c(X, j') - c_{ij'})_+ = f_i$  for  $i \notin X$ 
    Assign  $(j, t)$  to  $i$ ;  $X \leftarrow X \cup \{i\}$ ;  $D \leftarrow D \cup \{(j, t)\}$ 
   $t \leftarrow t + 1$ 

```

---

### 4.2 The Analysis

We first analyze the cost of the primal solution.

**Lemma 4.** *The cost of the solution produced by the algorithm is at most  $2 \sum_{(j,t) \in \mathcal{D}} v_{jt}$ .*

*Proof.* We show that both the service costs and the facility costs of the solution are each bounded above by  $\sum_{(j,t) \in \mathcal{D}} v_{jt}$ , which will give the lemma. Note that when a client  $(j, t)$  is assigned to a facility  $i$ , either it had made a bid on that facility and caused it to open, in which case  $(v_{jt} - c_{ij})_+ > 0$ , implying  $v_{jt} > c_{ij}$ , or  $i$  was already open and  $v_{jt} = c_{ij}$ . In either case,  $v_{jt}$  is at least the service cost  $c_{ij}$ .

To bound the facility costs  $\sum_{i \in X} f_i$ , we note that when we opened a facility  $i$ , its cost was equal to the sum of the bids on that facility. We show that for a given client  $(j, t)$ , the sum of its accepted bids is at most  $v_{jt}$ , which implies that  $\sum_{i \in X} f_i \leq \sum_{(j,t) \in \mathcal{D}} v_{jt}$ . In particular, we show that once the bid of client  $(j, t)$  is used to open some facility  $i$ , all its other outstanding bids are reduced by the amount bid towards  $i$ ; since it bids at most  $v_{jt}$  towards any facility at any point in time, this is sufficient to prove the claim. Consider two facilities,  $i$  and  $i'$ . Before  $(j, t)$  is connected to any facility, it bids  $(v_{jt} - c_{ij})_+$  towards the first and  $(v_{jt} - c_{i'j})_+$  towards the latter. If facility  $i$  is opened and  $(j, t)$  is assigned to  $i$ , then  $(j, t)$  reduces its bid for  $i'$  to  $(c(X, j) - c_{i'j})_+ = (c_{ij} - c_{i'j})_+$ ; that is, it reduced its bid for  $i'$  by  $v_{jt} - c_{ij}$ , exactly the bid accepted for opening facility  $i$ . Similarly, if  $(j, t)$  is assigned to some facility, it bids towards unopened facilities  $i$  and  $i'$   $(c(X, j) - c_{ij})_+$  and  $(c(X, j) - c_{i'j})_+$  respectively. If its bid towards  $i$  is accepted, and  $i$  is opened, then it must have been the case that  $(c(X, j) - c_{ij})_+ > 0$ , so that  $i$  is closer to  $j$  than any other facility in  $X$ . Once  $i$  is opened, it is added to  $X$  so that the bid to  $i'$  becomes  $(c(X \cup i, j) - c_{i'j})_+ = (c_{ij} - c_{i'j})_+$ . The bid towards  $i'$  is reduced by  $c(X, j) - c_{ij}$ , exactly the bid accepted towards  $i$ .  $\square$

The following lemma will be useful in bounding bids towards facilities in the remainder of the proof.

**Lemma 5.** *Consider clients  $(j, t)$  and  $(l, t')$ , such that  $j$  is considered before  $l$ , so that we increase the dual of  $(j, t)$  before that of  $(l, t')$ . Then for  $X$  open when we increase  $v_{l t'}$ , for any facility  $i$ ,  $c(X, j) - c_{ij} \geq v_{l t'} - c_{il} - 2c_{ij}$ .*

Let  $H_n$  be the  $n$ th harmonic number  $1 + \frac{1}{2} + \dots + \frac{1}{n}$ . Our goal is to show that for  $\alpha = 1/2H_n$ , and for any facility  $i$ ,  $\sum_{(j,t) \in \mathcal{D}} (\alpha v_{jt} - c_{ij})_+ \leq f_i$ . Thus scaling down the dual solution  $v$  by  $2H_n$  gives a feasible solution to the dual program in Section 2. To prove this, we show the following lemma.

**Lemma 6.** *For any  $S \subseteq \mathcal{D}$  and any facility  $i$ ,  $\sum_{(j,t) \in S} (\alpha v_{jt} - c_{ij}) \leq f_i$ .*

*Proof.* For ease of exposition, we let  $S = \{1, 2, \dots, p\}$ , dropping the pair notation for clients, with the understanding that we increase the dual variables for the clients in the order of the indices. Consider any  $l \in S$ . At the point in time at which we increase the dual for  $l$ , the total bids for facility  $i$  are  $(v_l - c_{il})_+ + \sum_{j < l} (c(X, j) - c_{ij})_+ \leq f_i$ .

So we have

$$\begin{aligned} f_i &\geq (v_l - c_{il})_+ + \sum_{j < l} (c(X, j) - c_{ij})_+ \geq (v_l - c_{il}) + \sum_{j < l} (c(X, j) - c_{ij}) \\ &\geq (v_l - c_{il}) + \sum_{j < l} (v_l - c_{il} - 2c_{ij}) \geq l(v_l - c_{il}) - 2 \sum_{j < l} c_{ij}, \end{aligned}$$

where the penultimate inequality follows from Lemma 5. Dividing both sides by  $l$  we get  $\frac{1}{l} f_i \geq (v_l - c_{il}) - \frac{2}{l} \sum_{j < l} c_{ij}$ . Adding the inequality for all  $l$  in  $S$  and observing that  $\sum_{l=1}^p \frac{2}{l} \sum_{j < l} c_{ij} = \sum_{l=1}^p 2c_{il}(H_p - H_l)$ , gives

$$\begin{aligned}
 H_p f_i &\geq \sum_{l=1}^p (v_l - c_{il}) - \sum_{l=1}^p 2c_{il}(H_p - H_l) \\
 &= \sum_{l=1}^p (v_l - 2c_{il}H_p) + \sum_{l=1}^p 2c_{il} \left( H_l - \frac{1}{2} \right) \geq \sum_{l=1}^p (v_l - 2c_{il}H_p).
 \end{aligned}$$

Dividing by  $2H_p$  we get  $\sum_{l=1}^p \left( \frac{v_l}{2H_p} - c_{il} \right) \leq \frac{f_i}{2} \leq f_i$ . Since  $H_p \leq H_n$ , then the lemma statement holds. □

**Corollary 1.**  *$\alpha v$  is a dual feasible solution.*

The theorem below follows from the previous discussion.

**Theorem 2.** *Fotakis’ algorithm gives a  $4H_n$ -competitive algorithm for the on-line facility location problem.*

## 5 An Algorithm for Online Facility Leasing

Here, we modify Fotakis’ algorithm to give a  $O(K \log n)$ -competitive algorithm for online facility leasing problem. Our algorithm constructs a dual infeasible solution and a primal feasible integral solution that costs no more than  $(K + 1)$  times the dual objective function. Then we show that by scaling down the duals by a factor of  $O(\log n)$  we get a dual feasible solution yielding the required competitive ratio. The main difference of our algorithm is that clients use their dual variables to bid on the  $K$  different lease types at the same time, and bids are reduced on leases of length  $k$  only as a lease of type  $k$  is opened. At a very high level, the factor of  $O(K)$  comes from clients contributing to each lease type simultaneously, and the  $O(\log n)$  from the underlying online facility location problem.

### 5.1 The Algorithm

First, following Meyerson [7] and Anthony and Gupta [1], it will be useful to change the leases available to be somewhat more structured.

**Lemma 7 (Lemma 2 [1]).** *Given an instance  $I$  of the facility leasing problem, it can be converted into another instance  $I'$  of the facility leasing problem such that leases of type  $k$  are only available at times  $t$  divisible by  $l_k$  such that any solution to  $I$  can be converted into a solution to  $I'$  that costs no more than twice as much.*

In particular, this implies an  $O(\alpha)$ -competitive algorithm for the instances with leases of this structure is  $O(\alpha)$ -competitive for the general problem. From here on, we assume that leases are structured in this way.

Following the ideas of the previous section, we maintain a set of dual variables  $v_{jt}$  for each  $j \in D_t$ . We also maintain a set of facilities  $X$  opened so far and set of

facilities  $X^k$  of lease length  $k$  opened so far. Every client  $(j, t)$  that has arrived at some prior time bids  $(\min[v_{jt}, c(X^k, j)] - c_{ij})$  towards a facility  $(i, k, t')$  if  $t \in I_{t'}^k$ . We maintain the invariant that the sum of the bids of all the clients seen so far to a facility is no more than the facility's opening cost.

When a new client arrives, we increase the dual of that client until either an the total bids towards some unopened facility is equal to its facility cost, or the client's dual is equal to its cost to receive service from an already open facility, whichever occurs earlier. In first case, we open the facility, connect the client to it, and reduce the bids of all clients contributing to this facility to other unopened facilities of the same lease type  $k$  by the bid value of the client. In the second case, the client is connected to the open facility. We then repeat these steps for each of the arriving clients.

---

**Algorithm 2.** Online Leasing Algorithm

---

```

 $X \leftarrow \emptyset; X^k \leftarrow \emptyset; D \leftarrow \emptyset; t \leftarrow 0$ 
While true
  For each  $j \in D_t$ 
    Increase  $v_{jt}$  until  $v_{jt} = c_{ij}$  for  $(i, k, \hat{t}) \in X$  and  $t \in I_{\hat{t}}^k$ 
      or  $(v_{jt} - c_{ij})_+ + \sum_{(j', t') \in D, t' \in I_{\hat{t}}^k} (\min[v_{j't'}, c(X^k, j')] - c_{ij'})_+ = f_i^k$ 
      for  $(i, k, \hat{t}) \notin X$  and  $t \in I_{\hat{t}}^k$ 
    Assign  $(j, t)$  to  $(i, k, \hat{t})$ ;  $X \leftarrow X \cup \{(i, k, \hat{t})\}$ ;
     $X^k \leftarrow X^k \cup \{(i, k, \hat{t})\}$ ;  $D \leftarrow D \cup \{(j, t)\}$ ;
   $t \leftarrow t + 1$ 

```

---

**5.2 The Analysis**

Here we show that for  $\alpha = \frac{1}{2(H_n + 1)}$ ,  $\sum_{(j,t) \in D, t \in I_{t'}^k} (\alpha v_{jt} - c_{ij})_+ \leq f_i^k$  for any facility  $(i, k, t')$ .

**Lemma 8.** For any  $S \subseteq D$  and any facility  $(i, k, t')$ ,  $\sum_{(j,t) \in S, t \in I_{t'}^k} (\alpha v_{jt} - c_{ij}) \leq f_i^k$ .

*Proof.* Without loss of generality we assume that  $S$  consists only of clients  $(j, t)$  such that  $t \in I_{t'}^k$  as any other client  $(j, t) \in S$  such that  $t \notin I_{t'}^k$  will not contribute to the summation. We say a client  $(j, t)$  is connected to an open facility  $(i, k, \hat{t})$  if  $v_{jt} \geq c_{ij}$  and  $t \in I_{\hat{t}}^k$ . Let  $S = \{1, 2, \dots, p\}$  where we index the clients in  $S$  according to the order in which they became connected to a open facility of lease length  $k$  in the algorithm. If there are clients which are connected to a facility of lease type  $k$  in the same iteration of the algorithm, then we order them according to non-increasing order of their  $v_{jt} - c_{ij}$ . Note that we have dropped the pair notation for the clients for ease of exposition.

Consider any client  $l \in S$  which was connected to a open facility of lease level  $k$  at some point in time. Let  $X^k$  be the set of facilities of  $k^{th}$  lease level that are open at the start of the iteration in which  $l$  is first connected to an open facility

of  $k^{th}$  level lease. Let  $h \leq l$  be the first client in the ordering that was connected to a  $k^{th}$  level lease in the same iteration as the client  $l$ . Consider the invariant for the facility  $(i, k, t')$  at the time the client  $l$  became connected to a  $k^{th}$  lease type.

$$\begin{aligned}
 f_i^k &\geq \sum_{h \leq j \leq l} (v_j - c_{ij})_+ + \sum_{j < h} (c(X^k, j) - c_{ij})_+ \\
 &\geq \sum_{h \leq j \leq l} (v_j - c_{ij}) + \sum_{j < h} (c(X^k, j) - c_{ij}) \\
 &\geq (l - h + 1)(v_l - c_{il}) + \sum_{j < h} (v_l - c_{il} - 2c_{ij}) \tag{2}
 \end{aligned}$$

$$\geq l(v_l - c_{il}) - 2 \sum_{j < l} c_{ij} \tag{3}$$

Inequality (2) follows from the following claim and our ordering of the set  $S$ ; we know for all  $j$  with  $h \leq j \leq l$ ,  $v_j - c_{ij} \geq v_l - c_{il}$ .

*Claim.* For any client  $j \in S$  which is connected to a  $k^{th}$  level facility in an earlier iteration than when  $l \in S$  is connected to a  $k^{th}$  level facility,  $c(X^k, j) - c_{ij} \geq v_l - c_{il} - 2c_{ij}$ .

Dividing Inequality (3) by  $l$  we get  $\frac{1}{l} f_i^k \geq (v_l - c_{il}) - \frac{2}{l} \sum_{j < l} c_{ij}$ . Let  $q$  be the least index in  $S$  such that for all  $j > q$ , client  $j$  does not connect to any  $k^{th}$  level lease while the lease for  $(i, k, t')$  is available. Then we know that the invariant at the end of the lease gives  $\sum_{j \in S: j > q} (v_j - c_{ij}) \leq f_i^k$ . Adding the inequality above and the prior inequality for all  $l \in \{1, \dots, q\}$ , we get

$$\begin{aligned}
 (H_q + 1)f_i &\geq \sum_{l=1}^p (v_l - c_{il}) - \sum_{l=1}^q \frac{2}{l} \sum_{j < l} c_{ij} \\
 &= \sum_{l=1}^p (v_l - c_{il}) - \sum_{l=1}^q 2c_{il}(H_q - H_l) \\
 &\geq \sum_{l=1}^p (v_l - 2c_{il}H_q) + \sum_{l=1}^q 2c_{il} \left( H_l - \frac{1}{2} \right) \\
 &\geq \sum_{l=1}^p (v_l - 2c_{il}H_q) \geq \sum_{l=1}^p (v_l - 2c_{il}(H_q + 1)).
 \end{aligned}$$

Dividing by  $2(H_q + 1)$  we get  $\sum_{l=1}^p \left( \frac{v_l}{2(H_q + 1)} - c_{il} \right) \leq \frac{f_i^k}{2} \leq f_i^k$  which proves dual feasibility for  $\frac{v_l}{2(H_n + 1)}$  since  $H_q \leq H_n$ . □

**Lemma 9.** *The cost of opening facilities in  $X$  is no more than  $K$  times the sum of the duals of the clients.*

**Lemma 10.** *The sum of the connection costs of the clients to its assigned facilities is no more than the sum of the duals.*



**Corollary 2.**  $\alpha v$  is a dual feasible solution for  $\alpha = 1/(2H_n + 1)$ .

**Theorem 3.** The online facility leasing algorithm is a  $O(K \log n)$ -competitive algorithm.

## 6 Conclusion

The most interesting open question arising from this work is whether the  $O(K \log n)$  factor is nearly tight for a deterministic algorithm. It is possible that the  $\Omega(K)$  deterministic bound of Meyerson [7] for the parking permit problem and the  $\Omega(\frac{\log n}{\log \log n})$  bound of Fotakis [3] for the online facility location problem can be combined to give a deterministic  $\Omega(K \frac{\log n}{\log \log n})$  lower bound on the competitive ratio of the facility leasing problem. Potentially, however, much better bounds are possible. In particular, it would be interesting to consider randomized online algorithms for the problem.

## References

1. Anthony, B.M., Gupta, A.: Infrastructure leasing problems. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 424–438. Springer, Heidelberg (2007)
2. Fotakis, D.: A primal-dual algorithm for online non-uniform facility location. *Journal of Discrete Algorithms* 5, 141–148 (2007)
3. Fotakis, D.: On the competitive ratio for online facility location. *Algorithmica* (2007); Published online October 24, 2007. Previous version in ICALP 2003
4. Jain, K., Mahdian, M., Markakis, E., Saberi, A., Vazirani, V.: Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM* 50, 795–824 (2003)
5. Jain, K., Vazirani, V.V.: Approximation algorithms for metric facility location and  $k$ -median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM* 48, 274–296 (2001)
6. Meyerson, A.: Online facility location. In: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, pp. 426–431 (2001)
7. Meyerson, A.: The parking permit problem. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 274–282 (2005)
8. Odyssey Logistics, <http://www.odysseylogistics.com>
9. Swamy, C., Shmoys, D.B.: Sampling-based approximation algorithms for multi-stage stochastic optimization. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 357–366 (2005)

# Importance Sampling via Load-Balanced Facility Location

Aaron Archer and Shankar Krishnan

AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932  
{aarcher, krishnas}@research.att.com

**Abstract.** In this paper, we consider the problem of “importance sampling” from a high dynamic range image, motivated by a computer graphics problem called *image-based lighting*. Image-based lighting is a method to light a scene by using real-world images as part of a 3D environment. Intuitively, the sampling problem reduces to finding representative points from the image such that they have higher density in regions of high intensity (or energy) and low density in regions of low intensity (or energy).

We formulate this task as a facility location problem where the facility costs are a function of the demand served. In particular, we aim to encourage load balance amongst the facilities by using V-shaped facility costs that achieve a minimum at the “ideal” level of demand. We call this the *load-balanced facility location* problem, and it is a generalization of the uncapacitated facility location problem with uniform facility costs. We develop a primal-dual approximation algorithm for this problem, and analyze its approximation ratio using dual fitting and factor-revealing linear programs. We also give some experimental results from applying our algorithm to instances derived from real high dynamic range images.

## 1 Introduction

This paper introduces a discrete optimization model to address a problem arising in computer graphics. The clustering problem we propose, a variant of facility location, is NP-hard, so we design an approximation algorithm to attack it. This approach is unusual for computer graphics, where most methods that are proposed are heuristics that do not come with any sort of provable guarantees on the quality of the solutions they produce.

Suppose that we wish to synthetically render a scene that is being illuminated by multiple directional light sources. The challenge here is to accurately represent the reflections of light off of the surfaces in the scene. The way that light scatters off of a surface depends on physical properties of the surface material and on the directions from which incident light rays are striking the surface. By knowing these two things, we can compute how much light would be scattered in the direction of the viewer’s eye, and hence how we should render the surface.

One typically assumes that all of the light sources are infinitely far away, so that the incident illumination is identical at every point in space throughout the scene, modulo occlusions and reflections from objects inside the scene. Therefore, to capture all relevant information about the lighting, it is enough to obtain a single *light probe image*,

a spherical map of the intensity of light coming from each direction. From this spherical map, we can then compute the light incident on any given point via ray tracing, accounting for other elements of the scene that might occlude the direct lighting from certain directions, and also for single or multiple reflections from surfaces in the scene. This lighting method is called *image-based lighting* in computer graphics—that is, a method to light a scene by using real-world imagery as part of a 3D environment. The resolution of the light probe image is much greater than the number of rays we can typically afford to trace, and therefore we desire to sample a much smaller number of rays and trace only those. It is this sampling problem that we address in this paper.

Intuitively, we would expect to obtain the most faithful rendering if we sample many rays in regions of high intensity (or energy), and allocate relatively few in regions of low intensity (or energy). In other words, the quality of approximating an image-based lighting environment as a finite number of point lights is increased if the light positions are chosen to follow the distribution of the incident illumination. This is called the *importance sampling* problem. Debevec [4] suggests the following guideline. Suppose that the total intensity summed over all pixels in the light probe image is  $I$ , and we have a budget of  $B$  light rays (i.e., pixels from the light probe image) to sample. According to Debevec, subdividing the image into  $B$  regions of equal intensity and choosing a single light source in each region (center or centroid) should be sufficient. While he suggests a particular heuristic for defining the regions and placing a light source in each, we can interpret his strategy as follows: we should try to sample our  $B$  light rays such that when we compute the Voronoi diagram of our chosen rays and sum the intensities of the pixels in each Voronoi region, the total intensity is split as evenly as possible amongst the regions, i.e., each sum is as close as possible to  $I/B$ .

We cast this ray sampling task as a type of facility location problem. Every pixel is a client with demand equal to its intensity, and every pixel is also a potential facility. We wish to open a set of facilities and assign each unit of demand to some open facility so as to minimize the sum of two types of costs.<sup>1</sup> The first is a connection cost for each unit of demand, proportional to the distance between that demand and the facility that serves it. We assume that these distances form a metric. The second cost is incurred by each open facility, and is a function of the amount of demand served by that facility. The facility cost will be a “V” shape, described by three parameters. The first is a *target capacity*  $U$ , which denotes the ideal amount of demand that the facility should serve. Second, there is a fixed cost  $F_0$  to open the facility. Third, there is a penalty  $\Lambda$  for each unit by which the total demand served by the facility deviates from the ideal capacity  $U$ . Thus, if the facility serves  $D$  units of demand, the facility cost would be  $F_0 + \Lambda|D - U|$ . If  $D > U$  we say that the facility is *overflow*; if  $D < U$  we say that it is *underfull*. We denote by  $d_{ij}$  the connection cost of routing one unit of demand at client  $j$  to facility  $i$ . We call this the *load-balanced facility location problem* (LBFL). Notice that when  $\Lambda = 0$ , our problem reduces to the uncapacitated facility location problem with uniform facility costs, which is NP-hard.

Let us examine how we could apply this model to obtain a good solution for our ray sampling problem. Here, the connection costs are given by the great circle distance on the sphere. These costs encourage each demand to be routed to its Voronoi center, while

<sup>1</sup> Note that we allow a single client to split its demand amongst multiple facilities.

the facility costs encourage the facilities to be placed such that each serves roughly  $U$  units of demand. Clearly, we should set  $U = I/B$ . This leaves us two knobs to turn,  $F_0$  and  $\Lambda$ , in order to achieve our twin goals of opening roughly  $B$  facilities and splitting demand evenly amongst them. Consider a demand point  $j$ , whose nearest open facility (i.e., Voronoi center) is  $i$ . This demand can be routed to some other facility  $i'$  only if  $i$  is overfull,  $i'$  is underfull, and  $d_{i'j} < d_{ij} + 2\Lambda$ , because in this case the increased connection cost is overwhelmed by saving a penalty of  $\Lambda$  on each of the two facility costs. Thus, small values of  $\Lambda$  force the clusters to look more like Voronoi cells, while large values of  $\Lambda$  push the cluster sizes towards  $U$ . Setting  $F_0$  to be sufficiently high will cause only one facility to be opened, which will result in high connection costs and a highly overfull facility. As we decrease  $F_0$  while keeping  $\Lambda$  fixed, the number of facilities will increase, in order to decrease both the connection cost and the penalty for overfullness. Thus, our choice of  $\Lambda$  expresses a tradeoff between how well our clusters should resemble Voronoi cells and how much the demand should be balanced among them. Fixing a value for  $\Lambda$ , we can adjust  $F_0$  so as to open roughly  $B$  facilities.

There are other ways that we could have modeled ray sampling as a discrete optimization problem in order to satisfy Debevec's criterion of roughly balanced Voronoi cells. The most obvious would be to explicitly request  $B$  samples such that the vector of demands contained in the  $B$  Voronoi regions they induce is as close as possible to the vector  $(I/B, \dots, I/B)$  under some norm. Another option would be to formulate it as a facility location problem with a hard constraint that the total demand served by each facility must fall inside some window surrounding  $I/B$ . We chose our formulation because it seemed to be the most tractable.

**Related Work.** Facility location problems have been a major focus of study in operations research since the 1960s, and in theoretical computer science since the 1990s. Many variants of this problem have been studied. The most popular one is the *uncapacitated* version (UFL), where each facility  $i$  pays a fixed cost for being opened, and can then serve an unlimited amount of demand for no extra cost. See [14, pp. 22-25] for an excellent summary of much of this work and a long list of references.

Since most variants of the problem are NP-hard, they have inspired a large body of work developing approximation algorithms for them, particularly for the uncapacitated version. An *r-approximation algorithm* is an algorithm that runs in polynomial time, and is guaranteed to return a solution of value at most  $r$  times the optimum;  $r$  is called the *approximation ratio*. The uncapacitated facility location problem has provided fertile ground for developing a large range of approximation techniques, including LP rounding, randomized rounding, pipage rounding, cost-scaling, greedy augmentation, local search, and primal-dual methods, including dual fitting and factor-revealing linear programs (FRLPs). Again, see [14] for references. The current best approximation factor, due to Mahdian, Ye and Zhang [16], is 1.52 and uses a combination of cost-scaling and dual fitting, analyzed using FRLPs. Regarding hardness, Guha and Khuller [7] proved that no 1.463-approximation algorithm exists unless  $NP \subseteq \text{DTIME}[n^{O(\log \log n)}]$ .

Several papers [9, 15, 5] have given approximation algorithms for facility location problems with facility costs that vary with the demand served, but all of these assume that these costs are increasing with the demand. In contrast, our costs are V-shaped, so

they first decrease then increase. This turns out to drastically change the structure of the problem. It also causes standard local search techniques such as those in [15,5] to break. These algorithms bound the connection cost by considering a sequence of moves that add all the optimal facilities to the current solution and re-route all demand to them. In our problem, shedding demand from a facility can actually *increase* its cost, so this bounding technique breaks.

Svitkina [17] recently gave a  $(558 + \epsilon)$ -approximation algorithm for the *lower-bounded facility location problem*, which is identical to UFL, except that each facility has a hard lower bound on the amount of demand it must serve. This model was previously introduced by [8,13]. It is not appropriate for our desired application, because it does not give the opportunity to penalize for overloading a facility, so the solutions are not load-balanced. For our graphics application, it might be reasonable to use a model that imposes both upper and lower bounds on the demands served. However, an optimal solution for this model would tend to push the demands toward the bounds, so we prefer our formulation since it encourages true even splitting.

We use the technique of dual fitting, which involves constructing a dual solution to an LP relaxation of our problem along with an integer primal solution in such a way that the dual exactly pays for the primal, but is infeasible and hence does not give a valid lower bound. However, it can be scaled down by some factor  $\gamma$  to give a valid lower bound, resulting in an approximation guarantee of  $\gamma$ . One uses an FRLP to find a value of  $\gamma$  that will work for all instances. This technique goes back to Chvatal's analysis of the greedy algorithm for set cover [3], and was formalized by Jain et al. [11], who applied it to UFL. Mahdian, Ye and Zhang then used it to obtain the current best approximation factor for that problem [16], and it has been successfully applied to other combinatorial optimization problems as well [2,10,12]. FRLPs were first used explicitly by Goemans and Kleinberg [6] (although not in the dual fitting framework) to analyze their algorithm for the minimum latency problem. Our work extends the techniques and applicability of dual fitting and FRLPs.

**Our Results.** We develop an approximation algorithm for the LBFL problem where the approximation factor depends on a parameter  $\lambda = \Delta U / F_0$ , but is a constant with respect to the size of the input. We analyze the approximation ratio using the technique of dual fitting and FRLPs. This means that the analysis of our approximation algorithm boils down to analyzing the optimal solutions of an infinite family of linear programs (LPs). This involves identifying patterns in the optimal dual solutions to these LPs, so that we can establish a valid upper bound on the optimal value of each LP in the infinite family.

Like the FRLPs previously used for UFL [11,16], our FRLP has quadratically many "triangle inequality" constraints. Using a clever trick based on the max flow / min cut theorem, we reduce this to a linear number (Theorem 1). This is important for three reasons. First, the new, smaller LP and its dual are much simpler and help highlight the structure of their solutions. Second, the smaller size allows us to solve larger FRLPs faster, which is useful when exploring the solution structure. Both of these features are important because a detailed understanding of the structure of optimal solutions to the FRLP is needed to prove upper bounds on the approximation ratio of our algorithm.

Third, Theorem 1 can be applied to [11,16] as well, allowing a simpler derivation of their results and possibly aiding any future work that uses their framework.

In Jain et al.'s analysis for UFL, the collection of FRLPs that requires analysis is only a single-parameter family, corresponding to the possible values of the total demand served by a single facility. Our collection is indexed also by  $U$ ,  $\lambda$  and another parameter  $k^*$ , so it is a 4-parameter family. This is a significantly more complex beast, but we are able to tame it. The approximation guarantee that we derive depends on  $\lambda$ , which is inherent in our approach because the LP we use to get lower bounds on  $OPT$  has an integrality gap of  $\Omega(\lambda)$ .

Finally, we applied our algorithm to instances arising from real images, and report the results.

## 2 The Model

The input to our problem is a set of *facilities*  $\mathcal{F}$ , a set of *clients* (or *demand points*)  $\mathcal{D}$ , a metric  $d$  on  $\mathcal{F} \cup \mathcal{D}$  where  $d_{ij}$  denotes the distance from facility  $i$  to demand point  $j$ , a fixed facility cost  $F_0$ , an ideal capacity  $U$  and a penalty parameter  $\Lambda$ . These last three parametrize a facility cost function  $F_{F_0, U, \Lambda}(D) = F_0 + \Lambda|D - U|$  for  $D \geq 0$ . We assume for simplicity of exposition that  $U$  is an integer and every client has unit demand, but everything can still be made to work without this assumption. While  $F_0$ ,  $U$  and  $\Lambda$  are the most natural parameters from a modeling perspective, for our analysis it will be more convenient to rescale the facility costs. Let  $\lambda = \Lambda U / F_0$ ,  $p = F_0 / U$ , and  $f_{U, \lambda}(D) = U + \lambda|D - U|$ . Then  $F_{F_0, U, \Lambda}(D) = p f_{U, \lambda}(D)$ . We will ordinarily omit the subscripts  $U$  and  $\lambda$ , and write the facility cost as  $pf(D)$ .

The goal is to open a subset of facilities  $\mathcal{O} \subseteq \mathcal{F}$  and select an assignment  $\phi : \mathcal{D} \rightarrow \mathcal{O}$  of demands to open facilities so as to minimize the quantity

$$\sum_{j \in \mathcal{D}} d_{\phi(j)j} + \sum_{i \in \mathcal{O}} pf(|j : \phi(j) = i|),$$

which is the sum of the connection costs of each client to the facility that serves it, plus the sum of the facility costs for each facility to serve the amount of demand routed to it. Notice that for any fixed set of facilities, the optimal assignment of demands can be computed using min-cost flow, so (by integrality of flows) if we were to allow ourselves to split a client's unit demand over multiple facilities, it would not lower the cost.

In much of our analysis, we will have to distinguish between the two cases  $\lambda < 1$  and  $\lambda > 1$ , owing to a qualitative difference between the two. For a facility serving exactly the ideal demand  $U$ , the facility cost per unit demand is  $p$ , while the over- or underfullness penalty is  $p\lambda$ . Thus, for  $\lambda > 1$ , the cost per unit demand achieves a minimum when the facility is exactly full, whereas for  $\lambda < 1$ , the cost per unit demand continues to decrease even once the facility is overfull.

## 3 Our Algorithm and Analytic Framework

Let us define a *star*  $(i, S)$  to be a facility  $i \in \mathcal{F}$ , along with a collection of demand points  $S \subseteq \mathcal{D}$  that it might serve. Every solution to our facility location problem can be

viewed as a partition of the demand points by stars, where the center of the star is an open facility and the facility serves exactly the demand points in  $S$ . This problem can be modeled by an integer program whose linear relaxation is shown below, along with its LP dual. We use  $c_{(i,S)} = \sum_{j \in S} d_{ij} + pf(|S|)$  to denote the cost of the star  $(i, S)$ .

$$\begin{aligned}
 \min \quad & \sum_{(i,S)} c_{(i,S)} x_{(i,S)} \\
 \text{s.t.} \quad & \sum_{\substack{i \in \mathcal{F}, \\ S \ni j}} x_{(i,S)} = 1, \quad \forall j \in \mathcal{D} \\
 & \sum_S x_{(i,S)} \leq 1, \quad \forall i \in \mathcal{F} \\
 & 0 \leq x_{(i,S)} \leq 1, \quad \forall (i, S)
 \end{aligned}
 \quad
 \begin{aligned}
 \max \quad & \sum_{j \in \mathcal{D}} \alpha_j - \sum_{i \in \mathcal{F}} \mu_i \\
 \text{s.t.} \quad & \sum_{j \in S} \alpha_j - \mu_i \leq c_{(i,S)}, \quad \forall (i, S) \\
 & \mu_i \geq 0, \quad \forall i \in \mathcal{F}.
 \end{aligned} \tag{1}$$

We use a dual-fitting algorithm to construct a feasible primal solution along with an infeasible solution to the dual. Our algorithm will fix all  $\mu_i$  variables to zero and never change them. Our primal and dual solutions will have the same objective function value, and we will exhibit some  $\gamma$  such that multiplying our dual solution by  $\frac{1}{\gamma}$ , makes it feasible and hence provides a lower bound on  $OPT$ . Thus, our primal solution is a  $\gamma$ -approximation. The main thrust of our work will be to prove an upper bound on the maximum  $\gamma$  that could ever be necessary to make the scaled dual solution feasible.

Conceptually, our algorithm starts all clients  $j$  with  $\alpha_j = 0$ , and raises these dual variables uniformly until some star  $(i, S)$  becomes tight. At this point, we open  $i$ , connect all clients in  $S$  to it, and freeze their dual variables. We then proceed as before, except that we consider only unfrozen clients when computing the next star to open. For an already-open facility  $i$  that is currently underfull, if any client  $j$  achieves  $\alpha_j = d_{ij} - p\lambda$ , then we connect  $j$  to  $i$  and freeze it. If  $i$  is already full or overfull, then  $j$  must achieve  $\alpha_j = d_{ij} + p\lambda$  to connect this way. We always raise the unfrozen dual variables uniformly until we first hit one of these new star or direct connection events, at which point we process the event, then continue until all demands are frozen.

This is the most natural algorithm to consider, given that we are using LP (1) and aim to use dual fitting for the analysis; it is very similar to one in (11) for UFL. It can be implemented to run in  $O(m \log m)$  time, where  $m = |\mathcal{F}||\mathcal{D}|$ . The implementation is similar to that in (11), so we omit the details.

**Deriving the Factor-Revealing LP.** By construction, the  $\sum_j \alpha_j$  resulting from the algorithm exactly pays for the primal. Since the smallest acceptable value of the scale factor  $\gamma$  is

$$\max_{(i,S)} \frac{\sum_{j \in S} \alpha_j}{c_{(i,S)}}, \tag{2}$$

we need to find an upper bound on this quantity. Notice that the max is taken over *all possible stars*, not just the ones that the algorithm actually opens. Consider the star  $(i, S)$  that maximizes (2), and let  $k = |S|$ . We rename the clients in  $S$  as  $1, \dots, k$  such that  $\alpha_1 \leq \dots \leq \alpha_k$ , and abbreviate  $d_{ij}$  as  $d_j$ . Then  $c_{(i,S)} = \sum_{j \in S} d_j + pf(k)$ . Since scaling down  $p$  and all  $d_{ij}$  uniformly causes the algorithm to behave the same but with



the  $\alpha$  scaled, we can assume WLOG that  $c_{(i,S)} = 1$ . Thus, our goal reduces to finding the largest possible value that  $\sum_{j \in S} \alpha_j$  can be, given that the algorithm imposes the following constraints.

Facility  $i$  was opened during the time interval  $[\alpha_{k^*}, \alpha_{k^*+1})$ , for some index  $k^*$  (where  $k^* = 0$  if  $i$  was opened before time  $\alpha_1$ , and  $k^* = k$  if  $i$  was opened after  $\alpha_k$  or never). We claim that the following linear program, denoted  $P(k, U, \lambda, k^*)$ , provides an upper bound on the approximation ratio for this instance. The constraints are labeled by the names we will give to their corresponding variables in the dual of this LP.

$$\begin{aligned}
 &\text{maximize} && \sum_{j=1}^k \alpha_j \\
 &\text{subject to} && \sum_{j=1}^k d_j + pf(k) = 1, && (\nu) \\
 &&& \alpha_j \leq \alpha_l + d_l + d_j + 2\lambda p, && \forall 1 \leq l < j \leq k && (\text{Tri}_{lj}) \\
 &&& |S|\alpha_j \leq \sum_{g \in S} d_g + pf(|S|), && \forall j \leq k^*, S \subseteq [k] && (\text{Star}_S) \\
 &&& && \text{s.t. } \min S = j && \\
 &&& \alpha_j \leq d_j + \lambda p && \forall j > k^* && (\text{DC}_j) \\
 &&& \alpha_j \leq \alpha_{j+1} && \forall 1 \leq j < k && (\sigma_j) \\
 &&& \alpha, d, p \geq 0
 \end{aligned}$$

We need to justify constraints  $(\text{Tri}_{lj})$ ,  $(\text{DC}_j)$ , and  $(\text{Star}_S)$ . For  $(\text{DC}_j)$ , recall that facility  $i$  is already open before time  $\alpha_j$ , so once  $j$  pays for its connection cost modulo the  $\pm \lambda p$  penalty/discount, it will connect to  $i$  as a singleton.

For  $(\text{Tri}_{lj})$ , consider the situation faced by client  $j$  just before it connects to some facility at time  $\alpha_j$ . Demand  $l < j$  connected to some facility  $i'$  at time  $\alpha_l$ . Since  $l$  received a discount of at most  $\lambda p$ , we know  $\alpha_l \geq d_{i'l} - \lambda p$ . To connect to  $i'$  later as a singleton,  $j$  would be charged at most a  $\lambda p$  penalty. Thus, we have

$$\alpha_j \leq d_{i'j} + \lambda p \leq d_{i'l} + d_{il} + d_{ij} + \lambda p \leq (\alpha_l + \lambda p) + d_l + d_j + \lambda p,$$

from the triangle inequality and the previous lower bound on  $\alpha_l$ . Thus,  $(\text{Tri}_{lj})$  is valid.

For  $(\text{Star}_S)$ , consider the point in time just before demand  $j$  connects to some facility. Each demand  $g \geq j$  is still active at this point, and at this time all active demands have dual variable equal to  $\alpha_j$ . Since  $i$  is not yet open at this point, one possible star is the one centered at  $i$  and serving exactly these demands. Thus, the sum of their dual variables cannot exceed the cost of opening this star, since then the star would have been opened strictly before time  $\alpha_j$ . Constraint  $(\text{Star}_S)$  expresses this.

Let us denote the optimal value of  $P(k, U, \lambda, k^*)$  by  $\rho(k, U, \lambda, k^*)$ . Then our goal is to compute the quantity  $\rho(\lambda) = \sup_{k,U,k^*} \rho(k, U, \lambda, k^*)$ , because this is an upper bound on the violation factor of any possible star in the dual solution generated by our algorithm, and is hence a valid approximation guarantee. Our approximation bound must depend on  $\lambda$  since a simple example shows that the integrality gap of  $(\text{II})$  is  $\Omega(\lambda)$ .



### 4 Analysis of the Factor-Revealing LP

In this section, we study the structure of  $P(k, U, \lambda, k^*)$  in depth, so we can understand what optimal solutions to this LP look like. In particular, we would like to construct a near-optimal solution to its dual, which will give a good upper bound on  $\rho(\lambda)$ . We start by simplifying the LP itself. We then show that the worst case LP is when  $U = k^* = k$  and  $k \rightarrow \infty$ . Finally, we construct a dual solution for this worst case to establish an upper bound on  $\rho(\lambda)$ .

**Simplifying  $P(k, U, \lambda, k^*)$ .**  $P(k, U, \lambda, k^*)$  has  $\binom{k}{2}$   $(Tri_{l_j})$  constraints, and it turns out that many of these are redundant. By considering the feasibility system consisting of only these constraints, taking its dual, performing a transformation based on the max flow-min cut theorem and taking the dual again we can derive the following theorem (whose proof we defer to the full paper because of space limitations):

**Theorem 1.** *The non-negative variables  $\alpha, d, p$  satisfy all the  $(Tri_{l_j})$  constraints iff there exist  $\omega_1, \dots, \omega_k \geq 0, t \in \mathbb{R}$  s.t.*

$$\alpha_j \geq t - d_j - \lambda p + \sum_{g>j} \omega_g, \quad \forall 1 \leq j < k \tag{TG_j}$$

$$\alpha_j \leq t + d_j + \lambda p + \sum_{g \geq j} \omega_g, \quad \forall 1 < j \leq k. \tag{TL_j}$$

Therefore, if we add the  $\omega$  variables to  $P(k, U, \lambda, k^*)$  and replace the  $(Tri_{l_j})$  constraints with the  $(TG_j)$  and  $(TL_j)$  constraints, we end up with an equivalent LP, which we denote  $\bar{P}(k, U, \lambda, k^*)$ . The dual of this LP is  $\bar{D}(k, U, \lambda, k^*)$ :

minimize  $\nu$

subject to

$$\left. \begin{aligned} & TL_j - TG_j + DC_j + \sigma_j - \sigma_{j-1} \\ & + \sum_{S: \min S=j} |S| \text{Star}_S \end{aligned} \right\} \geq 1 \quad \forall 1 \leq j \leq k \tag{\alpha_j}$$

$$\nu \geq TL_j + TG_j + \sum_{S:j \in S} \text{Star}_S + DC_j \quad \forall 1 \leq j \leq k \tag{d_j}$$

$$f(k)\nu \geq \sum_S f(|S|)\text{Star}_S + \lambda \sum_{j=1}^k (TL_j + TG_j + DC_j) \tag{p}$$

$$\sum_{j=1}^k TG_j - \sum_{j=1}^k TL_j = 0 \tag{t}$$

$$\sum_{l:l < j} TG_l \geq \sum_{l:l \leq j} TL_l \quad \forall 1 \leq j \leq k \tag{\omega_j}$$

$$DC_j = 0 \ (j \leq k^*), \text{Star}_S = 0 \ (S \text{ s.t. } \min S > k^*)$$

$$\sigma_0 = TL_1 = TG_k = 0, TL, TG, \text{Star}, \sigma \geq 0$$

**Reducing to  $\rho(k, k, \lambda, k)$ ,  $k \rightarrow \infty$ .** We next proceed to show that  $U = k^* = k$ , with  $k \rightarrow \infty$  is the worst case for this LP (i.e., the value of the optimal solution is highest for this case).

**Theorem 2.** *For all  $k, U, \lambda, k^*$ , we have  $\rho(k, U, \lambda, k^*) \leq \rho(k, k, \lambda, k^*)$ .*

**Proof:** Fix any solution to  $\bar{D}(k, k, \lambda, k^*)$ . We show that this solution is also feasible for  $\bar{D}(k, U, \lambda, k^*)$  for all  $U$ , and the result follows. The only constraint in  $\bar{D}(k, U, \lambda, k^*)$  that depends on  $U$  is (p), since the functions  $f(\cdot)$  depend implicitly on  $U$ . Thus, we only need to show that these constraints are still satisfied when we change  $U$ .

Let  $s_l = \sum_{S:|S|=l} \text{Star}_S$ , and  $g(U) = f_U(k)\nu - \sum_l f_U(l)s_l - \lambda \sum_j (\text{TL}_j + \text{TG}_j + \text{DC}_j)$  (i.e.,  $g(U)$  is the slack in the (p) constraint). We know  $g(k) \geq 0$  and wish to show that  $g(U) \geq 0$  for all  $U$ . Recall that

$$f_U(l) = U + \lambda|l - U| = \begin{cases} (1 - \lambda)U + \lambda l, & \text{if } U \leq l \\ (1 + \lambda)U - \lambda l, & \text{if } U \geq l \end{cases} \tag{3}$$

Thus,  $\frac{\partial}{\partial U} f_U(l) = 1 + \lambda$  for all  $l \leq k \leq U$ . Moreover, for each fixed  $l$ ,  $f_U(l)$  is a convex function of  $U$ , so  $g(U)$  is concave on  $[0, k]$ .

First consider the case  $U \geq k$ . Differentiating gives  $g'(U) = (1 + \lambda)(\nu - \sum_l s_l)$ . But since  $f_k(k)\nu \geq \sum_l f_k(l)s_l + \lambda \sum_j (\text{TL}_j + \text{TG}_j + \text{DC}_j)$  and  $f_k(l)$  achieves its min at  $l = k$ , we have  $\nu \geq \sum_l s_l$ . Thus,  $g'(U) \geq 0$  for all  $U \geq k$ , so  $g(U) \geq 0$  as well.

Now consider the case  $U \leq k$ . Since  $g$  is concave on  $[0, k]$  and  $g(k) \geq 0$ , it is enough to show that  $g(0) \geq 0$ , which simplifies to  $k\nu \geq \sum_l l s_l + \sum_j (\text{TL}_j + \text{TG}_j + \text{DC}_j)$ . Summing all the (d<sub>j</sub>) constraints gives precisely this. ■

**Theorem 3.** *For all  $k, \lambda, k^*$ , and every  $M \in \mathbb{N}$ ,  $\rho(k, k, \lambda, k^*) \leq \rho(Mk, Mk, \lambda, Mk^*)$ .*

**Proof:** For simplicity, we prove the theorem for  $M = 2$ . The proof for general  $M$  is analogous. Given any feasible solution to  $\bar{P}(k, k, \lambda, k^*)$ , we will “double” it to obtain a feasible solution to  $\bar{P}(2k, 2k, \lambda, 2k^*)$  of equal value, and the result follows.

Let  $(\alpha, d, p, t, \omega)$  be our solution to  $\bar{P}(k, k, \lambda, k^*)$ . We propose a solution  $(\bar{\alpha}, \bar{d}, \bar{p}, \bar{t}, \bar{\omega})$  to  $\bar{P}(2k, 2k, \lambda, k^*)$ . Let  $\bar{p} = p/2$  and, for each  $j \in [k]$ , let  $\bar{\alpha}_{2j-1} = \bar{\alpha}_{2j} = \alpha_j/2$ ,  $\bar{d}_{2j-1} = \bar{d}_{2j} = d_j/2$ . By construction, the objective value is still the same, and constraints (v), (DC<sub>j</sub>), (σ<sub>j</sub>), are still satisfied. By Theorem II  $(\alpha, d, p)$  satisfies the (Tri<sub>l<sub>j</sub></sub>) constraints, so  $(\bar{\alpha}, \bar{d}, \bar{p})$  does too. By Theorem III there exist  $\bar{\omega}$  and  $\bar{t}$  such that the new solution satisfies (TG<sub>j</sub>) and (TL<sub>j</sub>). The (Star<sub>S</sub>) constraints are a bit more subtle.

Given any  $\bar{S} \subseteq [2k]$ , there exist sets  $O, E \subseteq [k]$  s.t.  $\bar{S} = \{2j - 1 : j \in O\} \cup \{2j : j \in E\}$ . Summing constraints (Star<sub>O</sub>) and (Star<sub>E</sub>) for the original solution gives

$$\begin{aligned} |O|\alpha_{\min O} + |E|\alpha_{\min E} &\leq \sum_{g \in O} d_g + \sum_{g \in E} d_g + pf_k(|O|) + pf_k(|E|) \\ (|O| + |E|)\alpha_{\min(O \cup E)} &\leq 2 \sum_{g \in \bar{S}} \bar{d}_g + 2\bar{p}(k + \lambda(k - |O|) + k + \lambda(k - |E|)) \tag{4} \\ 2|\bar{S}|\alpha_{\min \bar{S}} &\leq 2 \sum_{g \in \bar{S}} \bar{d}_g + 2\bar{p}f_{2k}(|\bar{S}|), \end{aligned}$$

where (4) follows from (σ<sub>j</sub>). This last inequality is simply twice (Star<sub>S</sub>). ■

**Theorem 4.** For all  $k, \lambda, k^*$ , we have  $\rho(k, k, \lambda, k^*) \leq \rho(k, k, \lambda, k)$ .

**Proof sketch:** In  $\bar{D}(k, k, \lambda, k^*)$ , when we change  $k^*$  from  $k$  to something smaller, we lose some (Star<sub>S</sub>) constraints but gain some (DC<sub>j</sub>) constraints. Our goal is to show that this makes the LP tighter. Combining the (DC<sub>j</sub>) and (σ<sub>j</sub>) constraints yields  $\alpha_j \leq \min_{g \geq j} d_g + p\lambda$ , for  $j > k^*$ . This constraint implies most of the (Star<sub>S</sub>) constraints. Fixing  $S \subseteq \{k^* + 1, \dots, k\}$ , with  $j = \min S$ , we have

$$|S|\alpha_j \leq |S| \min_{g \geq j} d_g + |S|p\lambda \leq \sum_{g \in S} d_g + |S|p\lambda$$

This implies the (Star<sub>S</sub>) constraint as long as  $|S|\lambda \leq f(|S|) = k + \lambda(k - |S|)$ , that is,  $|S| \leq \frac{(1+\lambda)k}{2\lambda}$ . This is always the case as long as either  $\lambda \leq 1$ , or  $\lambda \geq 1$  and  $k^* \geq \frac{\lambda-1}{2\lambda}k$ . The proof of the remaining case (i.e.,  $\lambda \geq 1, k^* < \frac{\lambda-1}{2\lambda}k$ ) is much more involved, so we defer it to the full version of paper. ■

Putting Theorems 2, 3 and 4 together yields:

**Corollary 5.**  $\rho(\lambda) = \lim_{k \rightarrow \infty} \rho(k, k, \lambda, k)$ .

**Bounding  $\rho(k, k, \lambda, k)$  with a Canonical Dual.** Now that we have reduced our problem to analyzing  $\lim_{k \rightarrow \infty} \rho(k, k, \lambda, k)$ , we can use an LP solver to help us further explore the structure of the optimal solutions to  $\bar{D}(k, k, \lambda, k)$  and  $\bar{P}(k, k, \lambda, k)$ . Theorem 3 suggests the optimal solutions will be roughly scale-invariant, meaning that if we express all the indices as fractions of  $k$  and scale down the primal variables and the dual Star<sub>S</sub> variables by a factor of  $k$ , the solutions for large  $k$  should look very similar as  $k$  varies. We solved a bunch of these LPs with CPLEX, which confirmed this intuition.

**Table 1.**  $\rho(k, k, \lambda, k)$  for different values of  $k$  and  $\lambda$  (generated by CPLEX), along with the best upper bound on  $\rho(\lambda)$  that we can prove analytically

$k$	$\lambda = 0.0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
50	1.7979	1.9499	2.0948	2.2310	2.4684	3.0212	3.5694
100	1.8064	1.9588	2.1039	2.2400	2.4844	3.0427	3.5967
200	1.8107	1.9632	2.1084	2.2445	2.4925	3.0535	3.6106
400	1.8128	1.9655	2.1107	2.2467	2.4966	3.0590	3.6175
proof	1.8608	2.0396	2.1719	2.2500	2.5007	3.0645	3.62438

Table 1 shows the computed value of  $\rho(k, k, \lambda, k)$  for a range of  $\lambda$  and  $k$  values, along with the best upper bound we can prove for  $\rho(\lambda)$  using the method outlined below. For fixed  $\lambda$ , we see that the approximation bound  $\rho(k, k, \lambda, k)$  appears to be converging fairly rapidly. Since the full details of proving the bounds are quite hairy, we defer them to the full version of the paper, but here we sketch the steps involved.

Our overall goal is to create a “canonical” solution to  $\bar{D}(k, k, \lambda, k)$  which can be “projected” onto a feasible dual solution for any given value of  $k$ , thereby yielding an

upper bound on  $\rho(k, k, \lambda, k)$ . Because of Corollary 5 we care about the quality of this bound only for large  $k$ .

We first establish which variables in  $\bar{P}(k, k, \lambda, k)$  and  $\bar{D}(k, k, \lambda, k)$  are non-zero. We can readily deduce some of the patterns just by looking at the LPs. For instance, since we expect  $p$  to be non-zero in an optimal solution, there is no way for both (TG<sub>*j*</sub>) and (TL<sub>*j*</sub>) to be tight for the same  $j$ . Thus, by complementary slackness, TG<sub>*j*</sub> and TL<sub>*j*</sub> should never both be non-zero. Inspection of the optimal solutions output by CPLEX reveals further structure. For instance, when  $\lambda \geq 0.8022$ , the solutions adhere to the following pattern. All  $\alpha_j$  are positive, and all  $\sigma_j$  and  $\omega_j$  are zero. The only Star<sub>*S*</sub> variables that are non-zero are for tail stars, i.e., ones of the form  $S_j = \{j, \dots, k\}$ . There exists a partition of the indices  $\{1, \dots, k\}$  into three intervals,  $I_1 = \{1, \dots, b_1k\}$ ,  $I_2 = \{b_1k + 1, \dots, b_2k\}$ ,  $I_3 = \{b_2k + 1, \dots, k\}$ , where  $0 \leq b_1 \leq b_2 \leq 1$  are roughly constant with respect to  $k$ , but depend on  $\lambda$ . These intervals determine the non-zero pattern of the remaining variables, as follows. For  $j \in I_1$ ,  $d_j$ , TG<sub>*j*</sub>, and Star<sub>*S<sub>j</sub>*</sub> are non-zero, while Star<sub>*S<sub>j</sub>*</sub> is also non-zero on  $I_2$ , and TL<sub>*j*</sub> is non-zero on  $I_3$ . All other primal and dual variables are zero.

Once we have identified the pattern of non-zeros in the primal and dual, we use the complementary slackness conditions to determine which dual constraints should be tight. We then solve this system of linear equations to determine the non-zeros in the dual solution, up to a dependence on a small handful of parameters. Doing this for the pattern described above yields the following solution in terms of the parameters  $b_1, b_2, c$ : Star<sub>*S<sub>j</sub>*</sub> =  $\frac{c}{k}$  on  $I_1$  and  $\frac{1}{|S_j|}$  on  $I_2$ ; TG<sub>*j*</sub> =  $|S_j|\frac{c}{k} - 1$  on  $I_1$ ; TL<sub>*j*</sub> = 1 on  $I_3$ . Any choice of these parameters defines a canonical dual solution (TG, TL, Star<sub>*S*</sub>,  $\sigma_j, \nu$ ). Since the (d<sub>*j*</sub>) and (p) constraints are just lower bounds on  $\nu$ , as long as (α<sub>*j*</sub>), (u), (ω<sub>*j*</sub>) and the non-negativity constraints are satisfied, we can just set  $\nu$  to be the max of these lower bounds. Thus, any choice of the parameters that causes the dual to satisfy these constraints is valid. Treating the parameters as variables, we can solve a non-linear program to optimize them such that  $\nu$  is as small as possible. This whole process results in a proof of our upper bound on  $\rho(\lambda)$ , for any particular  $\lambda$ . When  $\lambda = 1$  for example, we get  $(b_1, b_2, c) = (0.2068, 0.5577, 3.5007)$ , yielding  $\nu = 2.5007$ . Fortunately, the behavior of  $\bar{D}(k, k, \lambda, k)$  is qualitatively the same over intervals of  $\lambda$  (e.g.,  $\lambda \geq 0.8022$ ), so we have to derive only a few of these “parameter-revealing” non-linear programs. However, to obtain the actual bound on  $\rho(\lambda)$ , we need to solve one of these non-linear programs for that particular  $\lambda$ . This is why we do not give an explicit formula for our approximation bound as a function of  $\lambda$ , although we can say that it is asymptotic to  $\lambda$ . Since a trivial example gives an integrality gap of  $\lambda/3$  for LP (II), our algorithm is within a constant of the best one can do if one uses LP (II) as the lower bound on *OPT*.

**Notes on Solving  $\bar{D}(k, U, \lambda, k^*)$ .** As we have shown, actually solving  $\bar{P}(k, U, \lambda, k^*)$  and  $\bar{D}(k, U, \lambda, k^*)$  computationally and inspecting the results was invaluable in deriving our approximation guarantees. It was also quite useful in guiding us to discover the theorems leading to Corollary 5. Thus, it is worth noting the multiple ways in which Theorem 1 helps us to solve these LPs efficiently and analyze them effectively, even though doing so is necessary only for the analysis of our algorithm, not for running the algorithm itself.

The transformation from  $P(k, U, \lambda, k^*)$  and  $D(k, U, \lambda, k^*)$  to  $\bar{P}(k, U, \lambda, k^*)$  and its dual  $\bar{D}(k, U, \lambda, k^*)$  allowed by Theorem 1 helps enormously in deciphering the form of the canonical dual solution.  $D(k, U, \lambda, k^*)$  has  $\binom{k}{2}$  Tri variables, which are highly redundant and thus cause a lot of degeneracy in the optimal solution of  $D(k, U, \lambda, k^*)$ , which makes it very difficult to pick out the relevant patterns in the Tri variables. Switching to  $\bar{D}(k, U, \lambda, k^*)$  greatly simplifies our task by getting rid of the degeneracy, and also by giving us many fewer variables to understand (just  $2k - 2$  TL and TG variables).

Since there are exponentially many  $\text{Star}_S$  variables, we solve  $\bar{P}(k, U, \lambda, k^*)$  using cutting planes (or equivalently, solve  $\bar{D}(k, U, \lambda, k^*)$  using column generation). It is easy to find the most violated  $\text{Star}_S$  constraint amongst all  $S$  s.t.  $\min S = j$ . We just sort clients  $g = j + 1, \dots, k$  by increasing  $d_g$ , and take all  $g$  for which  $\alpha_j - d_g + \lambda p > 0$ , provided there are at most  $U - 1$  of them. If there are more, we take the first  $U - 1$ , and in addition we take any  $g$  such that  $\alpha_j - d_g - \lambda p > 0$ . The most violated  $S$  consists of the clients we just identified, plus  $j$ .

We initialize the cutting plane method with the  $k$  tail stars (i.e.,  $S_1, \dots, S_k$ ). In our experience, the number of additional cuts generated was always less than  $3k$ . Thus, in practice, the LPs actually considered by the solver always have  $\Theta(k)$  variables and constraints. This highlights the next reason that Theorem 1 is so helpful: it reduces the size of our LPs from quadratic to linear in  $k$ . When trying to divine the asymptotic behavior of  $\rho(k, k, \lambda, k)$ , we want to push  $k$  as high as we can while still being able to solve  $\bar{D}(k, k, \lambda, k)$  reasonably quickly. Thus, the size and solution time of the LP really becomes an issue, even though we are using the LPs only to help us prove theorems.

Note that when  $U \geq k$ , we do not have to worry about stars of size larger than  $U$ . Thus, the separation oracle becomes easier since we need not sort the  $d_g$ . This also means that we could get rid of the  $(\text{Star}_S)$  constraints entirely by defining new variables  $C_{jt} \geq 0$  and adding the constraints  $C_{jt} \geq \alpha_t - d_j + \lambda p$  ( $\forall j \geq t$ ) and  $\sum_{j:j \geq t} C_{jt} \leq pU(1 + \lambda)$  ( $\forall t$ ). However, this is a bad idea because it again blows up the size of the LP from linear to quadratic in  $k$ .

## 5 Experimental Results

In this section, we describe some experimental results obtained from applying our algorithm to sample High Dynamic Range (HDR) images used in computer graphics for lighting scenes. We implemented our algorithm in C++. Its theoretical worst-case running time is  $O(m \log m)$  where  $m$  is the product of the number of demand points and the number of potential facilities. In this setting, the demand and facility sets are both identified with the pixels in the image. Hence  $m = (\# \text{ pixels})^2$ , which is prohibitively high even for moderate-sized images. However, in practice our implementation manages to avoid explicitly considering most (client, facility) pairs and hence runs much faster than  $O(m \log m)$ .

In order to further speed up the running time for our experiments, we need to reduce the cardinality of the facility set. We do so by generating some set of  $K \gg B$  pixels that we would expect to be good candidates for our final samples, and use these as  $\mathcal{F}$ . We then run our load-balanced facility location algorithm on this input to identify

roughly  $B$  of these  $K$  pixels as our final solution. Since we want our initial facility set  $\mathcal{F}$  to lie in regions of high intensity, a good solution to the  $K$ -median problem seems to be a natural attractive choice for  $\mathcal{F}$ <sup>2</sup>. We generate such an  $\mathcal{F}$  using an algorithm proposed by Arthur and Vassilvitskii [11]. Their algorithm chooses  $K$  random facilities in succession, according to a distribution that depends on the facilities already chosen. It gives an  $O(\log K)$ -approximate solution for  $K$ -median (in expectation). The algorithm, in addition to its theoretical guarantees, is very fast and simple to implement. For a budget of  $B$  samples, we set  $K$  to be either  $16B$  or  $32B$ .

Our algorithm has two parameters  $F_0$  (the facility cost at  $U$ ) and  $\Lambda$  (the penalty per unit demand for deviating from ideal capacity  $U$ ).  $\Lambda$  allows us to trade off Voronoi-ness against load balance. If  $\Lambda = 0$ , we get Voronoi cells. If  $\Lambda = \infty$ , we get perfect load balance. Given a fixed  $\Lambda$ ,  $F_0$  allows us to control how many facilities are opened. If  $F_0 = \infty$ , only one facility will be opened. When  $F_0 = 0$ , more will be opened. Adjusting these values gives us a way to control the number of open facilities as well as the load balance. We use the following strategy to set  $F_0$  and  $\Lambda$  so that we get roughly  $B$  approximately load-balanced facilities. We start with  $F_0 = 0$  and  $\Lambda = 0$ . This setting opens all facilities. We increase  $\Lambda$  until we open about  $2B$  facilities. We then increase  $F_0$  until the number of open facilities is close to  $B$ .

**Data Sets.** We use 5 high dynamic range images of various sizes to test our algorithm. They are named *Memorial*, *Rosette*, *Nave*, *Lamps* and *Tree*. Table 2 shows the statistics for the different input images.

**Table 2.** Statistics about the input images

Name	Image Size	# Fac. $ \mathcal{F} $	Budget $B$	Tot. Intensity $I$	$U$
Memorial	$512 \times 512$	2048	128	50889.8	397.6
Rosette	$256 \times 256$	4096	256	98443.0	384.5
Nave	$256 \times 256$	2048	64	191955.2	2999.3
Lamps	$256 \times 256$	1024	64	3104.0	48.5
Tree	$256 \times 256$	1024	64	11033.6	172.4

Table 3 shows the results of our algorithm on the input images. The first three columns indicate our settings for  $F_0$ ,  $\Lambda$  and  $\lambda = \Lambda U / F_0$ . The next four columns show the number of facilities opened by our algorithm, the total facility cost, the total connection cost and an upper bound on the approximation ratio for this particular instance. We derived our upper bound on the approximation ratio by computing (2), i.e., identifying the most violated star in the dual solution  $\alpha$  generated by our algorithm. The actual approximation ratio of this instance could be substantially better, since the lower bound generated by scaling down  $\alpha$  uniformly is likely to be fairly weak, compared to the bound one would obtain by actually solving LP (1). It may be feasible to solve this LP via column generation; we leave this to future work.

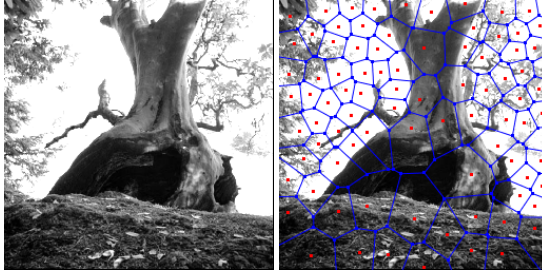
<sup>2</sup> The  $K$ -median problem is the same as UFL, except that there are no facility costs (only connection costs), and the number of open facilities is restricted to be  $K$ .

**Table 3.** Results from our algorithm

Name	$F_0$	$\Delta$	$\lambda$	# Open Fac.	Total Fac. Cost	Total Conn. Cost	Approx. Ratio
Memorial	45.0	1.3	11.4	134	6030.0	81529.9	5.15
Rosette	35.0	0.9	9.88	252	8820.0	116180.0	2.15
Nave	60.0	0.54	27.0	66	3960.0	197038.0	6.23
Lamps	90.0	0.3	0.16	72	6480.0	14471.8	1.32
Tree	122.5	0.37	0.52	88	10780.0	53787.2	1.38

**Table 4.** Evaluating our algorithm's output for load balance and Voronoianness

Name	# Fac. Open	Conn. Cost Ratio (ours/Voronoi)	Our Under/Over-full Demand (% of I)	Voronoi Under/Over-full Demand (% of I)
Memorial	134	1.0503	10.2655 / 5.5779	27.1197 / 22.4322
Rosette	252	1.0705	6.0096 / 7.5720	11.6977 / 13.2601
Nave	66	1.0547	11.8984 / 8.7734	17.9927 / 14.8676
Lamps	72	1.0626	41.3946 / 28.8821	39.1269 / 26.6140
Tree	88	1.0093	38.7969 / 1.2788	38.5549 / 1.0368

**Fig. 1.** Results of our algorithm on the *Tree* data set. On the left is the original image; on the right, we have superimposed our samples and their Voronoi cells.

Finally, to test the results with our original objective of sampling such that each open facility roughly serves equal demand and that most clients connect to their closest open facility, we compare the total connection cost of our output with that of the Voronoi cells for the for each facility. If these costs are close, that implies that most of the demand is met by the closest facility. We also list the total demand served by overfull and underfull facilities. If there is perfect load balance and we open exactly  $B$  (the budget) facilities, then they would be zero. Table 4 shows the details for each of the data sets. Figure 1 depicts the sampling produced by our algorithm on the *Tree* instance, along with the corresponding Voronoi cells.

## Acknowledgments

We thank David Applegate for sharing his expertise with AMPL.



## References

1. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. In: SODA, pp. 1027–1035 (2007)
2. Bansal, N., Fleischer, L., Kimbrel, T., Mahdian, M., Schieber, B., Sviridenko, M.: Further improvements in competitive guarantees for QoS buffering. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 196–207. Springer, Heidelberg (2004)
3. Chvatal, V.: A greedy heuristic for the set covering problem. *Math. Oper. Res.* 4, 233–235 (1979)
4. Debevec, P.: Image-based lighting. *IEEE Comput. Graph.* 22(2), 26–34 (2002)
5. Garg, N., Khandekar, R., Pandit, V.: Improved approximation for universal facility location. In: SODA, pp. 959–960 (2005)
6. Goemans, M.X., Kleinberg, J.M.: An improved approximation ratio for the minimum latency problem. *Math. Program.* 82, 111–124 (1998)
7. Guha, S., Khuller, S.: Greedy strikes back: improved facility location algorithms. *J. Algorithm* 31(1), 228–248 (1999)
8. Guha, S., Meyerson, A., Munagala, K.: Hierarchical placement and network design problems. In: FOCS, pp. 603–612 (2000)
9. Hajiaghayi, M.T., Mahdian, M., Mirrokni, V.S.: The facility location problem with general cost functions. *Networks* 42(6), 42–47 (2003)
10. Immorlica, N., Mahdian, M., Mirrokni, V.S.: Cycle cover with short cycles. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 641–653. Springer, Heidelberg (2005)
11. Jain, K., Mahdian, M., Markakis, E., Saberi, A., Vazirani, V.V.: Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM* 50(6), 795–824 (2003)
12. Jain, M.R.S.K., Mahdian, M.: Packing Steiner trees. In: SODA, pp. 266–274 (2003)
13. Karger, D.R., Minkoff, M.: Building Steiner trees with incomplete global knowledge. In: FOCS, pp. 613–623 (2000)
14. Mahdian, M.: Facility Location and the Analysis of Algorithms through Factor-Revealing Programs. PhD thesis, MIT, Cambridge, MA (June 2004)
15. Mahdian, M., Pál, M.: Universal facility location. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 409–421. Springer, Heidelberg (2003)
16. Mahdian, M., Ye, Y., Zhang, J.: Improved approximation algorithms for metric facility location problems. In: Jansen, K., Leonardi, S., Vazirani, V.V. (eds.) APPROX 2002. LNCS, vol. 2462, pp. 229–242. Springer, Heidelberg (2002)
17. Svitkina, Z.: Lower-bounded facility location. In: SODA, pp. 1154–1163 (2008)



# A Constant Approximation Algorithm for the *a priori* Traveling Salesman Problem

David Shmoys<sup>1,\*</sup> and Kunal Talwar<sup>2</sup>

<sup>1</sup> Cornell University, Ithaca NY 14853

shmoys@cs.cornell.edu

<sup>2</sup> Microsoft Research,

Silicon Valley Campus,

1065 L'Avenida,

Mountain View, CA 94043

kunal@microsoft.com

**Abstract.** One of the interesting recent developments in the design and analysis of approximation algorithms has been in the area of algorithms for discrete stochastic optimization problems. In this domain, one is given not just one input, but rather a probability distribution over inputs, and yet the aim is to design an algorithm that has provably good worst-case performance, that is, for *any* probability distribution over inputs, the objective function value of the solution found by the algorithm must be within a specified factor of the optimal value.

The *a priori* traveling salesman problem is a prime example of such a stochastic optimization problem. One starts with the standard traveling salesman problem (in which one wishes to find the shortest tour through a given set of points  $N$ ), and then considers the possibility that only a subset  $A$  of the entire set of points is active. The active set is given probabilistically; that is, there is a probability distribution over the subsets of  $N$ , which is given as part of the input. The aim is still to compute a tour through all points in  $N$ , but in order to evaluate its cost, we instead compute the expectation of the length of this tour after shortcutting it to include only those points in the active set  $A$  (where the expectation is computed with respect to the given probability distribution). The goal is to compute a “master tour” for which this expectation is minimized. This problem was introduced in the doctoral theses of Jaillet and Bertsimas, who gave asymptotic analyses when the distances between points in the input set are also given probabilistically.

In this paper, we restrict attention to the so-called “independent activation” model in which we assume that each point  $j$  is active with a given probability  $p_j$ , and that these independent random events. For this setting, we give a 8-approximation algorithm, a polynomial-time algorithm that computes a tour whose *a priori* TSP objective function value is guaranteed to be within a factor of 8 of optimal (and a randomized 4-approximation algorithm, which produces a tour of expected cost within a factor of 4 of optimal). This is the first constant approximation algorithm for this model.

---

\* Research supported partially by NSF grants CCR-0635121 & DMI-0500263.

## 1 Introduction

One of the interesting recent developments in the design and analysis of approximation algorithms has been in the area of algorithms for discrete stochastic optimization problems. In this domain, one is not given just one input, but rather a probability distribution over inputs, and yet the aim is to design an algorithm that has provably good worst-case performance, that is, for *any* probability distribution over inputs, the objective function value of the solution found by the algorithm is within a specified factor of optimal value.

The *a priori traveling salesman problem* is a prime example of such a stochastic optimization problem. This is a stochastic generalization of the (standard) traveling salesman problem (TSP). In the traditional deterministic problem, one is given a set  $N$  of  $n$  points in a metric space; that is, for each pair of points one is given the distance between them, with the assumption that these distances satisfy the triangle inequality (i.e., for each triple of points,  $i$ ,  $j$ , and  $k$ , the distance between  $i$  and  $k$ , is at most the sum of the distances between  $i$  and  $j$ , and between  $j$  and  $k$ ). The aim is to find the tour (or cyclic permutation) that contains all of these points exactly once, such that the total distance traversed is minimized. In the *a priori* TSP, one is also given a probability distribution  $\Pi$  over subsets  $A \subseteq N$ ; this models the fact that, in advance, one is not certain of the “currently active” points  $A$  that need to be included in the tour, and only has probabilistic information about this. However, given only this probabilistic information, one computes a “master tour”  $\tau$ , whose objective function value is the expected value (with respect to  $\Pi$ ) of the length of  $\tau$  after shortcutting it to include only those active points in  $A$ . If the distribution  $\Pi$  makes the full set  $N$  active with probability 1, then this is just the usual (deterministic) TSP (and hence we expect results no stronger than is known for that problem).

The *a priori* TSP was first studied in the doctoral theses of Jaillet [12] and Bertsimas [3], and their early work, which mostly focuses on asymptotic analysis when the points themselves are randomly selected (e.g., uniformly in the unit square), is nicely surveyed by Bertsimas, Jaillet, and Odoni [4]. If one is interested in polynomial-time algorithms, then one must be careful in the way that the distribution  $\Pi$  is given as part of the input. It would take an exponential amount of information to list the probability that each set  $A$  is active as part of the input; one simple workaround is to insist that only a polynomial (in  $n$ ) number of sets can be active with a positive probability, and this is called the *polynomial scenario* model. Alternatively, we can present the probability distribution by means of an oracle, a “black box” from which we can draw independent samples according the distribution  $\Pi$ ; the restriction that we are interested in polynomial-time algorithms implies that we are limited to a polynomial number of samples in this *black box* model.

In this paper, we restrict attention to the so-called *independent activation* model for  $\Pi$ ; for each  $j \in N$ , consider the random indicator variable,  $\mathbb{1}(j \in A)$ , which is 1 exactly when  $j \in A$ , and then we shall assume that these are

independent random variables, and that we are given, as part of the input, the probability  $p_j = \Pr[\mathbb{1}(j \in A) = 1]$ , for each  $j \in N$ . For this setting, we first give a randomized polynomial-time algorithm to compute a tour for the *a priori* TSP whose expected objective function value (with respect to the random choices of the algorithm) is guaranteed to be within a factor of 4 of optimal. We then derandomize the procedure to give a deterministic algorithm that is guaranteed to find a tour of cost within a factor of 8 of optimal. This is the first constant approximation algorithm for this model.

The strongest previous result was a randomized  $O(\log n)$ -approximation algorithm due to Schalekamp and Shmoys [5]. That result is significantly more general, since not only does it hold in the black box model, but in fact, it does not require *any* knowledge of the distribution  $\Pi$ .

This stochastic notion of the TSP is closely related to the notion of the *universal TSP*. In the universal TSP, there is also a notion of a “master tour” which is shortcut according to the current active set  $A$ . However, for this problem, we say that a tour  $\tau$  for  $N$  has a performance guarantee of  $\rho$  if, for every subset  $A \subseteq N$ , the tour  $\tau$  shortcut to  $A$  has length at most  $\rho$  times the optimal cost of a tour for the subset  $A$ . Clearly, a  $\rho$ -approximation algorithm for the universal TSP is a  $\rho$ -approximation algorithm for the *a priori* TSP with respect to *any* distribution. The universal TSP was introduced by Bartholdi and Platzman [6] in the context of an application for “meals on wheels”, and they gave an  $O(\log n)$  performance guarantee for points in the Euclidean plane. In contrast, Hajiaghayi, Kleinberg, and Leighton [7] proved that no tour can achieve a bound better than  $c(\log n / \log \log n)^{1/6}$  for some constant  $c > 0$ , even for points in the Euclidean plane. Thus, it is particularly natural to consider weaker models, such as the *a priori* TSP.

Our algorithm is extremely simple; in essence, we take one sample  $S$  from the active set distribution  $\Pi$ , build a minimum spanning tree on that subset, and then add each unselected point as a leaf in this tree by connecting it to its nearest neighbor in  $S$ . Then this spanning tree on  $N$  is converted to a tour through all points in  $N$ , by taking the standard “double tree” walk around the “outside” of the tree. In fact, one must be a bit more careful in handling the case in which the sample  $S$  is empty, and this will introduce a number of minor complications. This algorithm is a slight variant on an algorithm that was developed first by Gupta, Kumar, Pál, and Roughgarden [8] in a rather different context, that of the *rent-or-buy problem*, which is a deterministic network design problem in which one can either rent an edge  $e$  at a cost of  $c_e$  per use, or buy it for unlimited use at a cost of  $Mc_e$  (where  $M > 0$  is an input to the problem); this algorithm was later also the basis for an algorithm of Gupta, Pál, Ravi and Sinha [9] for the 2-stage (with recourse) stochastic Steiner tree problem. Williamson and van Zuylen [10] have derandomized the rent-or-buy algorithm, and their technique translates almost directly to our setting. Finally, we note that Garg, Gupta, Leonardi, and Sankowski [11,12], as a corollary of their work on the on-line stochastic Steiner tree, have independently obtained a similar constant approximation algorithm for the *a priori* TSP.

## 2 The Approximation Algorithm

### 2.1 Preliminaries

We start by introducing some notation needed to present our results. For the traveling salesman problem, the input consists of a finite set of points  $N = \{1, 2, \dots, n\}$  as well as the distance  $c(i, j)$  between each pair of distinct points  $i, j \in N$ ; we wish to find a tour  $\tau$ , given by a cyclic permutation of  $N$ , such that the length of the tour,  $c(\tau) = \sum_{i \in N} c(i, \tau(i))$  is minimized. We shall assume that the distances are symmetric (i.e.,  $c(j, k) = c(k, j)$  for each pair of points  $j, k \in N$ ) and satisfy the triangle inequality (i.e.,  $c(j, k) + c(k, \ell) \geq c(j, \ell)$  for each triple of points  $j, k, \ell \in N$ ).

For the *a priori* TSP, only a subset  $A \subseteq N$  will be “active”. For a tour  $\tau$  of  $N$ , we define  $\tau_A$  to be the tour “short-cut” to traverse only  $A$ ; more formally, if we let  $\tau^\ell$  be the  $\ell$ th power of  $\tau$ , (i.e.,  $\tau^1 = \tau$  and for each  $\ell > 1$ ,  $\tau^\ell = \tau(\tau^{\ell-1})$ ), then for each  $j \in A$ ,  $\tau_A(j) = \tau^\ell(j)$ , where  $\ell$  is the smallest positive integer such that  $\tau^\ell(j) \in A$  (unless  $A = \{j\}$ , in which case  $\tau^\ell(j)$  equals  $j$ ).

For the *a priori* TSP, the input consists of a set of points  $N$ , a distance function  $d$ , and a probability distribution  $\Pi$  specified over the subsets of  $N$ . Since we are working in the independent activation model, we can compute the probability  $\Pi(A)$  for each subset  $A \subseteq N$ ; that is,

$$\Pi(A) = \left( \prod_{j \in A} p_j \right) \left( \prod_{j \notin A} (1 - p_j) \right), \quad (1)$$

where  $p_j$  is the probability that point  $j$  is in the active set  $A$ . The goal now is to find a tour  $\tau$  that minimizes the *expected length* with respect to  $\Pi$  of the tour induced by  $\tau$ ; i.e., to minimize  $E_A[c(\tau_A)]$ . We shall let  $\tau^*$  denote an optimal solution, and let  $OPT$  denote the corresponding optimal value. It is important to note that  $\tau_A^*$  denotes the optimal tour shortcut to the subset of nodes  $A$ , which is not necessarily the optimal tour for this specific subset.

We will give an 8-approximation algorithm for this problem; that is, we will show how to compute a tour  $\tau$  such that  $E_A[c(\tau_A)] \leq 8 \cdot OPT$ . We first give a randomized algorithm that computes a tour  $\tau$  with the property that  $E[E_A[c(\tau)]] \leq 4 \cdot OPT$ , where the outer expectation in this expression is with respect to the random coin tosses used by the algorithm, but we will subsequently show how to derandomize this algorithm, while losing an additional factor of 2.

### 2.2 Special Case: $p_1 = 1$

As a warm-up, we will first consider a more structured case, where  $p_1 = 1$ .

The randomized algorithm is extremely simple. We first draw a sample  $S$  with respect to the underlying distribution  $\Pi$ . In other words, we choose a subset  $S$  by independently including every node  $j$  in  $S$  with probability  $p_j$ . We compute a minimum spanning tree on the subset  $S$ . Then, we extend this to be a spanning tree  $T$  on  $N$ , by adding, for each node  $j \notin S$ , the edge from  $j$  to its nearest

neighbor in  $S$ . (Note that this is well-defined, since we have ensured that  $S \neq \emptyset$  by setting  $p_1 = 1$ .) Let  $MST(S)$  denote the cost of the minimum spanning tree (MST) on  $S$  and for each  $j \neq 1$ , let  $D_j(S)$  denote the length of this edge between  $j$  and its nearest neighbor in  $S - \{j\}$ . (For notational convenience, let  $D_1(S) = 0$ ).

From this spanning tree  $T$ , it is well known (see, e.g., [13]) that we can construct a tour  $\tau$  of total length  $c(\tau)$  at most twice the total length of edges in the tree  $T$ ; we simply “walk around the outside of the tree,” or more formally, we construct an Eulerian multigraph by taking each edge twice, and then  $\tau$  is obtained from an Eulerian tour by shortcutting any node that is visited more than once.

To upper bound the objective function value of this tour  $\tau$ , we will focus on its tree representation; that is, for any subset  $A \subseteq N$ , the induced tour  $\tau_A$  can also be obtained by first considering the tree induced from  $T$  by the node set  $A$ , and then taking the analogous traversal of its doubled Eulerian tour.

In order to analyze the quality of this solution, we next present two obvious lower bounds on the length of an optimal solution.

**Fact 1.** For each subset  $A \subseteq N$ ,  $\tau_A^* \geq MST(A)$ .

**Fact 2.** For each subset  $A \subseteq N$ ,  $\tau_A^* \geq \sum_{j \in A} D_j(A)$ .

If we take expectations of both sides of these inequalities, we can thereby obtain lower bounds on the optimal value as well.

**Fact 3.**  $OPT \geq E_A[MST(A)]$ .

**Fact 4.**  $OPT \geq E_A[\sum_{j \in A} D_j(A)]$ .

Now let us analyze the cost of the (random) tree  $T$  generated by our algorithm. Focus on a particular active set  $A$ , and its contribution to the expected cost (with respect to  $\Pi$ ) of this solution. Recall that  $T$  consists of a spanning tree  $T_S$  on our sample  $S$ , plus additional edges that connect nodes  $j \notin S$  to this spanning tree. For a given set  $A$ , the induced tree  $T_A$  need not include all of the spanning tree  $T_S$ ; however, for computing an *upper bound* on the cost of  $T_A$ , we shall always include all of the edges of  $T_S$  in the induced solution. Given this, it is straightforward to see that we can upper bound the cost of  $T$  by

$$MST(S) + \sum_{j \neq 1} \mathbb{1}(j \in A) \mathbb{1}(j \notin S) D_j(S), \tag{2}$$

which is bounded above by

$$MST(S) + \sum_{j \neq 1} \mathbb{1}(j \in A) D_j(S). \tag{3}$$

Hence the expected cost (with respect to the random choices of our algorithm) of the tour  $\tau$  that we compute,

$$E_S[E_A[c(\tau_A)]] \leq 2 \left( E_S[MST(S)] + E_S[E_A[\sum_{j \neq 1} \mathbb{1}(j \in A) D_j(S)]] \right).$$

Since  $S$  is a random subset selected according to  $\Pi$ , the first term can be replaced by  $E_A[MST(A)]$ , which is, by Fact 3, at most  $OPT$ . For the second term, we apply linearity of expectation, and the fact that  $A$  and  $S$  are independent samples drawn according to  $\Pi$ , and hence

$$\begin{aligned}
 E_S[E_A[\sum_{j \neq 1} \mathbb{1}(j \in A)D_j(S)]] &= \sum_{j \neq 1} E_A[\mathbb{1}(j \in A)]E_S[D_j(S)] \\
 &= \sum_{j \neq 1} E_A[\mathbb{1}(j \in A)]E_A[D_j(A)],
 \end{aligned}
 \tag{4}$$

where again we have relied on the fact that the subsets  $S$  and  $A$  are both (independently) drawn according the distribution  $\Pi$ .

For each  $j \neq 1$ , the random variable  $D_j(A)$  denoting the distance from  $j$  to its nearest neighbor in  $A \setminus \{j\}$  is independent of  $\mathbb{1}(j \in A)$ . Thus we conclude that

$$\sum_{j \neq 1} E_A[\mathbb{1}(j \in A)]E_A[D_j(A)] = \sum_{j \neq 1} E_A[\mathbb{1}(j \in A)D_j(A)].
 \tag{5}$$

On the other hand, recall that by Fact 4,

$$OPT \geq E_A[\sum_j \mathbb{1}(j \in A)D_j(A)] = \sum_j E_A[\mathbb{1}(j \in A)D_j(A)].$$

Thus we can also bound the second term relative to  $OPT$ . Combining all of these pieces, we have shown that

$$E_S[E_A[c(\tau_A)]] \leq 2(OPT + OPT) = 4OPT.$$

### 2.3 General Case

Now we will relax the condition that there must exist one of the points with activation probability equal to 1. One might question where we take advantage of this restriction within the analysis given above. The way in which this gets used is most importantly in the description of the algorithm, in which there always is a set of non-empty points on which to build a spanning tree. Stated another way, for each point  $j$ , there is always a point in the sample of the algorithm in  $S - \{j\}$ . let  $\Pi'$  denote the probability distribution where  $\Pi'(A)$  is the probability that  $A$  is the active set (as drawn from  $\Pi$ ), conditioned on the event that  $|A| \geq 2$ . For a set  $A$  drawn according to  $\Pi$ , let  $\alpha$  denote the probability that  $|A| \geq 2$ . The following lemma implies that we can focus on  $\Pi'$  instead of  $\Pi$ .

**Lemma 5.** *For any tour  $\tau$ , its a priori TSP objective function value with respect to  $\Pi$  is exactly equal to  $\alpha$  times its objective function with respect to  $\Pi'$ .*

*Proof.* For each set  $A$  with  $|A| < 2$ , the cost of the short-cut of the tour  $\tau$  to the subset  $A$ ,  $c(\tau_A)$ , is equal to 0. For each subset  $A$  with  $|A| \geq 2$ , the probability that  $A$  is active with respect to  $\Pi$  is exactly equal to  $\alpha$  times the probability

that is active with respect to  $\Pi'$ . Now consider the objective function value of  $\tau$  with respect to  $\Pi$ , which is an expectation computed over the random choice of  $A$ . The subsets of size less than two contribute nothing (literally), and the rest contribute in total exactly  $\alpha$  times what they contribute to the analogous expected value for  $\Pi'$ . Hence, the claim is proved.

This lemma shows that our objective functions with respect to  $\Pi$  and  $\Pi'$  differ only by the same multiplicative factor (for all feasible solutions), and hence we get the following corollary.

**Corollary 6.** *Any (randomized)  $\rho$ -approximation algorithm for the *a priori* TSP with respect  $\Pi'$  is a (randomized)  $\rho$ -approximation algorithm with respect to  $\Pi$ .*

Now let us consider the extension of our algorithm and its analysis to  $\Pi'$ . If the algorithms draws a sample  $S$  according to  $\Pi'$ , we now have the property that for each  $j \in N$ , there must exist some element in  $S - \{j\}$ , and hence  $D_j(S)$  is well-defined. It is straightforward to see that much of the analysis of the our algorithm is unaffected by this change (though of course, the summation in the bound should now be done over all  $j \in N$ , not just those nodes  $j \neq 1$ ). The only significant change is that the random variables  $\mathbb{1}(j \notin S)$  and  $D_j(S)$  are no longer independent so that (5) no longer holds. Note that it would suffice to replace (5) by the inequality

$$\sum_j E_A[\mathbb{1}(j \in A)]E_A[D_j(A)] \leq \sum_j E_A[\mathbb{1}(j \in A)D_j(A)], \tag{6}$$

which, using basic properties of conditional expectation is equivalent to

$$\sum_j E_A[\mathbb{1}(j \in A)]E_A[D_j(A)] \leq \sum_j E_A[\mathbb{1}(j \in A)]E_A[D_j(A)|j \in A]. \tag{7}$$

If we prove that for every  $j \in N$ ,  $E_A[D_j(A)] \leq E_A[D_j(A)|j \in A]$ , then the remainder of the analysis carries over from the more specialized setting and the desired result follows. The modified algorithm yields a randomized 4-approximation algorithm.

**Lemma 7.** *For a random set  $A$  drawn according to the distribution  $\Pi'$ ,*

$$E_A[D_j(A)] \leq E_A[D_j(A)|j \in A].$$

*Proof.* The proof is based on the following two propositions:

**Proposition 8.** *For a random set  $A$  drawn according to the distribution  $\Pi'$ ,*

$$E_A[D_j(A)|j \notin A] = E_A[D_j(A)|(j \in A) \wedge (|A| \geq 3)].$$

*Proof.* Let  $\mathcal{S}$  be the family of sets of cardinality at least 2 that contain  $j$ , and let  $\bar{\mathcal{S}}$  be those that do not contain  $j$ . Finally, let  $\mathcal{S}_2$  denote the 2-element sets of  $\mathcal{S}$ .

As above, we know that there is a constant  $0 < \alpha \leq 1$ , such that for any subset  $A$  of cardinality at least 2,  $\Pi(A) = \alpha\Pi'(A)$ . Furthermore, we know how to compute  $\Pi$  (and  $\alpha$ ) from (II). There is a natural 1-1 correspondence  $f$  between the sets in  $\mathcal{S} - \mathcal{S}_2$  and  $\bar{\mathcal{S}}$ , by simply deleting  $j$ ; that is,  $f(S)$  maps  $S$  to  $S - \{j\}$ . But then,  $\Pi(S) = \frac{1-p_j}{p_j}\Pi(f(S))$ , and hence the same relation holds for  $\Pi'$  (provided  $|S| \geq 3$ ). Further,  $D_j(S) = D_j(f(S))$ , since each is the distance from  $j$  to  $S - \{j\}$ . In other words, the conditional expectation of  $D_j(S)$  for sets of size at least 3 containing  $j$  is exactly equal to the conditional expectation of  $D_j(S)$  for sets not containing  $j$  (where both expectations are with respect to the distribution  $\Pi'$ ).

**Proposition 9.** *For a random set  $A$  drawn according to the distribution  $\Pi'$ ,*

$$E_A[D_j(A)|(j \in A) \wedge (|A| \geq 3)] \leq E_A[D_j(A)|(j \in A)].$$

*Proof.* We prove instead that

$$E_A[D_j(A)|(j \in A)] \leq E_A[D_j(A)|(j \in A) \wedge (|A| = 2)],$$

which implies the lemma by the basic properties of conditional expectations.

By definition,  $D_j(\{j, k\}) = c(j, k)$ . Let  $\sigma$  be a permutation on  $N$  such that

$$0 = c(j, j) = c(j, \sigma(1)) \leq c(j, \sigma(2)) \leq \dots \leq c(j, \sigma(n)).$$

To compute either of these expectations, we need only consider the probability that each conditional distribution takes on each of these  $n - 1$  non-trivial values. We know that  $D_j(A) = c(j, \sigma(k))$  exactly when  $\sigma(\ell) \notin A$ ,  $\ell = 2, \dots, k - 1$ , and  $\sigma(k) \in A$ . Furthermore, for each distribution, the probability that any set is selected is proportional to the probability that it was selected in the original distribution  $\Pi$ , for which we know how to compute these probabilities exactly. Thus, for the first expectation, the probability that  $D_j(A) = c(j, \sigma(k))$  is proportional to  $p_{\sigma(k)} \cdot \prod_{\ell=2, \dots, k-1} (1 - p_{\sigma(\ell)})$ , whereas for the second, it is proportional to  $p_{\sigma(k)} \cdot \prod_{\ell \neq 1, k} (1 - p_{\sigma(\ell)})$  (with the same constant of proportionality). It is clear from these two expressions that the first probability dominates the second, so that for any  $k$ , it is more likely in the second case that  $D_j(S) \geq c(j, \sigma(k))$ . Consequently, the second expectation is at least the first, and the lemma is proved.

It follows that

$$E_A[D_j(A)|j \notin A] \leq E_A[D_j(A)|j \in A].$$

The lemma follows immediately since the unconditioned expectation is sandwiched in between these two values.

There is one final detail that should be mentioned. When we wish to select a sample from  $\Pi'$ , the most natural way is to repeatedly choose a sample from  $\Pi$ , until the resulting set has cardinality at least 2. If, for example, each of the values  $p_i = 2^{-n}$ , then this could take exponential time. However, one can easily compute the conditional probability that the two smallest indexed elements in



$S$  are  $i < j$  (for any such pair), and then the remaining elements can be selected by the independent selection rule from among  $\{j + 1, \dots, n\}$ . In this way, we can generate a sample according to  $\Pi'$  in polynomial time. Thus we have derived the following theorem.

**Theorem 10.** *There is a randomized algorithm that computes a solution  $\tau$  for the *a priori* TSP in polynomial time and has expected value (with respect to the random choices of the algorithm) no more than four times the optimal value.*

### 2.4 Derandomization

A stronger result would be to give a deterministic algorithm with the same performance guarantee. In fact, our algorithm in the simpler case with  $p_1 = 1$  is essentially identical to an earlier algorithm for the *rent-or-buy problem*, and we will exploit this connection in devising a deterministic analogue of Theorem 10. In this problem, one is given an undirected graph with edge costs  $c_e$  and an inflation factor  $M$ . One can rent edge  $e$  at a cost of  $c_e$  per transmission, or buy the edge  $e$  (and have unlimited capacity) for a total cost of  $Mc_e$ . A given subset of nodes are demand points that need to communicate with the root node 1. The aim is to decide which edges to buy and which to rent so that the total cost is minimized.

Gupta, Kumar, Pál, and Roughgarden [8] gave a very elegant approximation algorithm for this problem, which works as follows: choose a subset  $S$  by including node 1, and including each demand node  $i$ , independently, with probability  $1/M$ ; build a minimum spanning tree (or a near-optimal Steiner tree) on  $S$ , and these are the edges to buy; for each demand node  $i \notin S$ , rent the shortest edge from  $i$  to a node in  $S$  to serve that demand. Other than the fact that the inclusion probability for node  $i$  is  $1/M$ , rather than a specified parameter  $p_i$ , this is exactly the same algorithm as our approximation algorithm. Furthermore, Williamson and van Zuylen [10] have shown how to deterministically choose a tree for which the cost is at most the expected value of this randomized procedure. In fact, they gave a derandomization procedure for a slight restatement of this problem, called the *connected facility location problem*. In this problem, one selects some facilities to open, and then must connect them by a Steiner tree; for each node  $j$  not in the Steiner tree, one connects  $j$  to a node in  $i$  in the tree, by selecting the shortest such path (though of course, in a setting like ours in which the distances satisfy the triangle inequality, that path would be nothing more than a single edge).

It is quite straightforward to show that the natural generalization of the derandomization approach of Williamson and van Zuylen applies to our setting as well. Consider the bound that we used to analyze the performance of our algorithm, equation (2). This is the upper bound used in the expectation calculation for each active  $A$ , and so, for a particular choice of  $S$ , the overall bound that we obtain in this way is equal to twice

$$MST(S) + \sum_{j \neq 1} E_A[\mathbb{1}(j \in A)]\mathbb{1}(j \notin S)D_j(S) = MST(S) + \sum_{j \notin S} p_j D_j(S). \quad (8)$$

The essence of our proof is that if we select  $S$  at random, then this bound is good relative to the optimal cost. Thus, if we deterministically select a set  $S$  for which the actual cost (resulting from this upper bound on the expectation calculation) is no more than this expectation, then we have a deterministic approximation algorithm. It is not difficult to see that the problem of choosing a set  $S$  minimizing the expression (8) is an instance of the connected facility location problem, and hence using a (deterministic) 3.28-approximation (from [10]) to the problem gives a deterministic 13.12-approximation to the *a priori* TSP.

We next show how to use ideas from [10] directly to get a deterministic 8-approximation. The standard approach to derandomizing such an algorithm is the so-called *method of conditional expectations*; in this approach, one considers one random choice at a time, such as whether node  $j$  is in the set  $S$ , and computes the two conditional expectations, with this condition, and with its negation. While the conditional expectation of the second term in (8) is easy to compute, handling the first term seems rather difficult. Instead, Williamson & van Zuylen relied on a linear programming relaxation that captures enough of the structure of the tree optimization problem for which the method of conditional expectations can be applied. Thus, instead of computing the conditional expectation, we compute an upper bound (i.e., a pessimistic estimator) on this conditional expectation.

We first write the standard LP relaxation for the Steiner tree problem on a point set  $S$ , where  $\delta(U)$  denotes the set of edges  $ij$  such that  $i \in U, j \notin U$ :

$$\begin{aligned} \min \quad & \sum_{ij} c(i, j)y_{ij} \\ \text{(Steiner}(S)) \quad & \text{s.t. } \sum_{ij \in \delta(U)} y_{ij} \geq 1 \quad \forall U \subseteq N : S \cap U \neq \emptyset, 1 \notin U \\ & y_{ij} \geq 0 \end{aligned}$$

We next write a connected facility location type linear program:

$$\begin{aligned} \min \quad & B + C \\ \text{(CFL)} \quad & \text{s.t. } \sum_j x_{ij} = 1 \quad \forall i \in N \\ & \sum_{ij \in \delta(U)} z_{ij} \geq \sum_{j \in U} x_{ij} \quad \forall i \in N, \forall U \subseteq N : 1 \notin U \\ & B = \sum_{ij} c(i, j)z_{ij} \\ & C = \sum_{i \in N} p_i \sum_{j \in N} c(i, j)x_{ij} \\ & z_{ij}, x_{ij} \geq 0 \quad \forall i, j \in N \end{aligned}$$

Let  $(x^*, z^*, B^*, C^*)$  be an optimal solution to (CFL). We prove a sequence of lemmas:

**Lemma 11**

$$B^* + C^* \leq \frac{3}{2}OPT$$

*Proof.* Let  $OPT_A$  denote an optimal tour for instance  $A$ . For a sample  $A$ , let  $z^A$  be half the incidence vector of the optimal tour on  $A$  and for any  $i$ , let  $x_{ij}^A = 1$  if  $j$  is the nearest neighbor of  $i$  in  $A \setminus \{i\}$ . Clearly  $B^A = \frac{1}{2}OPT_A$  and  $C^A = \sum_i p_i D_j(A)$ . Note that the tour  $OPT_A$  crosses each set  $U \subseteq N : U \cap A \neq \emptyset, 1 \notin U$

at least twice so that the fractional value  $\sum_{ij \in \delta(U)} z_{ij}^A$  is at least one. It is then easy to check that  $(x^A, z^A, B^A, C^A)$  is a feasible solution for the linear program with objective function value  $\frac{1}{2}OPT_A + \sum_j p_j D_j(A)$ .

The vector  $E_A[(x^A, z^A, B^A, C^A)]$  is a convex combination of feasible points and hence is feasible itself. The objective function value for this solution, using Fact 4 and equation (5) is easily bounded by  $\frac{3}{2}OPT$ . Thus the optimal solution is no worse.

Given a set  $S \subseteq N$ , we now define a solution to  $\text{Steiner}(S)$  as follows:

$$\bar{y}_{ij}^S = z_{ij} + \mathbb{1}(i \in S)x_{ij}.$$

**Lemma 12.**  $\bar{y}^S$  is a feasible solution to  $\text{Steiner}(S)$ .

*Proof.* For any  $U \subseteq N : S \cap U \neq \emptyset, 1 \notin U$ , let  $i' \in S \cap U$ . Then:

$$\sum_{kl \in \delta(U)} \bar{y}_{kl}^S = \sum_{kl \in \delta(U)} z_{kl} + \mathbb{1}(k \in S)x_{kl} \geq \sum_{kl \in \delta(U)} z_{kl} + \sum_{l \notin U} x_{i'l} \geq \sum_{l \in U} x_{i'l} + \sum_{l \notin U} x_{i'l} = 1.$$

The claim follows.

With the above lemma in mind, we define  $\bar{c}_{ST}(S)$  to be the objective function value of this solution  $\bar{y}^S$  for  $\text{Steiner}(S)$ . One can check that the conditional expectation of  $\bar{c}_{ST}(S)$  can be directly evaluated. Moreover, since  $\bar{y}^S$  is a feasible solution to  $\text{Steiner}(S)$ , one can find a Steiner tree on  $S$  with cost at most  $2\bar{c}_{ST}(S)$ . Further, let  $c_R(S)$  denote  $\sum_j p_j D_j(S)$ .

The method of conditional expectation iterates through the points one by one and decides for each point whether or not to add it to  $S$ . We let  $P \subseteq N$  denote the points we decide to add to  $S$  and let  $\bar{P} \subseteq N$  denote the set of points we decide not to add. We define our estimator

$$Est(P, \bar{P}) = 2E[\bar{c}_{ST}(S) | P \subseteq S, \bar{P} \cap S = \emptyset] + E[c_R(S) | P \subseteq S, \bar{P} \cap S = \emptyset].$$

We start with  $P, \bar{P}$  empty, and maintain the invariant:

$$Est(P, \bar{P}) \leq 4OPT.$$

This holds in the beginning since  $2E[\bar{c}_{ST}(S)]$  is exactly  $2(B^* + C^*) \leq 3OPT$  by Lemma 11 and the second term is at most  $OPT$ , by Fact 4. Moreover, one can verify (see 10) that for  $i \notin P, \bar{P}$ ,

$$Est(P, \bar{P}) = p_i Est(P \cup \{i\}, \bar{P}) + (1 - p_i) Est(P, \bar{P} \cup \{i\}).$$

Thus the smaller of  $Est(P \cup \{i\}, \bar{P})$  and  $Est(P, \bar{P} \cup \{i\})$  is no larger than  $Est(P, \bar{P})$ . We can therefore add  $i$  to one of  $P$  or  $\bar{P}$  without violating the invariant.

At the end of the process,  $P \cup \bar{P} = N$  and we have a deterministic set  $S$  for which  $2\bar{c}_{ST}(S) + c_R(S) \leq 4OPT$ . Doubling the tree costs us another factor of two, and so we get an 8-approximation algorithm.

We have explained the derandomization in the context of the rooted variant. However, for the deterministic algorithm the distinction between these two cases essentially disappears. The probabilistic argument used to analyze the theorem proves that the expected cost over sets of size at least two is good (and can be bounded in the same way as previously). However, we can simply try all possible choices for the lexicographically smallest pair of nodes in  $S$ , choose the first as the root, and then continue the derandomization procedure for the remaining nodes. For each of the resulting at most  $n^2$  choices of  $S$ , we can evaluate the upper bound (8), and choose the best. The previous theorem ensures that the resulting solution will have the desired guarantee.

**Theorem 13.** *There is a deterministic algorithm that computes a solution  $\tau$  for the a priori TSP in polynomial time and has objective function value no more than eight times the optimal value.*

### 3 Conclusions

The constant 4 (or 8 for the deterministic case) can probably be optimized. There are at least two immediate possible sources of improvement. First, we are using the “double-tree” heuristic to compute the tour, rather than Christofides’s algorithm. This requires a slightly different analysis (since one must also bound the cost of the matching in terms of the *a priori* TSP). Furthermore, unlike in the result of Williamson & van Zuylen for rent-or-buy, we lose an extra factor of two relative to an LP relaxation. It seems likely that by adapting this technique to a stronger LP relaxation one can avoid this loss. On the other hand, our algorithm is competitive with respect to the expected value of the ex post *OPT*, which is a lower bound on the optimal cost for the problem. It is conceivable that one can get a better approximation guarantee if one uses a better lower bound on *OPT*. Finally, it is important to note that this algorithm is not a constant approximation algorithm in the black box model (or even the polynomial scenario model) for representing the probability distribution  $\Pi$ . It remains an interesting open question to provide a constant approximation algorithm in either of those two settings.

**Acknowledgments.** The authors would like to thank Alan Frieze, Shane Henderson, Frans Schalekamp, Anke van Zuylen, and David Williamson for helpful discussions.

### References

1. Jaillet, P.: Probabilistic traveling salesman problems. Technical Report 185, Operations Research Center, MIT (1985)
2. Jaillet, P.: A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research* 36, 929–936 (1988)

3. Bertsimas, D.: Probabilistic Combinatorial Optimization Problems. PhD thesis, MIT, Cambridge, Mass (1988)
4. Bertsimas, D.J., Jaillet, P., Odoni, A.R.: A priori optimization. *Operations Research* 38(6), 1019–1033 (1990)
5. Schalekamp, F., Shmoys, D.B.: Algorithms for the universal and *a priori* TSP. *Operations Research Letters* (in press, 2007)
6. Bartholdi III, J.J., Platzman, L.K.: An  $O(N \log N)$  planar travelling salesman heuristic based on spacefilling curves. *Operations Research Letters* 1(4), 121–125 (1981/82)
7. Hajiaghayi, M.T., Kleinberg, R., Leighton, T.: Improved lower and upper bounds for universal tsp in planar metrics. In: *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 649–658. ACM Press, New York (2006)
8. Gupta, A., Kumar, A., Pál, M., Roughgarden, T.: Approximation via cost-sharing: a simple approximation algorithm for the multicommodity rent-or-buy problem. In: *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, Silver Spring, MD, pp. 606–615. IEEE Computer Society, Los Alamitos (2003)
9. Gupta, A., Pál, M., Ravi, R., Sinha, A.: Boosted sampling: approximation algorithms for stochastic optimization. In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pp. 265–274 (2004)
10. Williamson, D.P., van Zuylen, A.: A simpler and better derandomization of an approximation algorithm for single source rent-or-buy. *Operations Research Letters* 35, 707–712 (2007)
11. Garg, N., Gupta, A., Leonardi, S., Sankowski, P.: Private communication (2007)
12. Garg, N., Gupta, A., Leonardi, S., Sankowski, P.: Stochastic analyses for online combinatorial optimization problems. In: *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 942–951. ACM and SIAM, New York and Philadelphia (2008)
13. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds.): *The Traveling Salesman Problem*. Wiley, Chichester (1985)

# New Geometry-Inspired Relaxations and Algorithms for the Metric Steiner Tree Problem<sup>\*</sup>

Deeparnab Chakrabarty<sup>1</sup>, Nikhil R. Devanur<sup>2</sup>, and Vijay V. Vazirani<sup>1</sup>

<sup>1</sup> College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280  
{[deepc](mailto:deepc@gatech.edu),[vazirani](mailto:vazirani@gatech.edu)}@cc.gatech.edu

<sup>2</sup> Toyota Technological Institute  
[nikhil@tti-c.org](mailto:nikhil@tti-c.org)

**Abstract.** Determining the integrality gap of the bidirected cut relaxation for the metric Steiner tree problem, and exploiting it algorithmically, is a long-standing open problem. We use geometry to define an LP whose dual is equivalent to this relaxation. This opens up the possibility of using the primal-dual schema in a geometric setting for designing an algorithm for this problem.

Using this approach, we obtain a  $4/3$  factor algorithm and integrality gap bound for the case of quasi-bipartite graphs; the previous best being  $3/2$  [RV99]. We also obtain a factor  $\sqrt{2}$  strongly polynomial algorithm for this class of graphs.

A key difficulty experienced by researchers in working with the bidirected cut relaxation was that any reasonable dual growth procedure produces extremely unwieldy dual solutions. A new algorithmic idea helps finesse this difficulty – that of reducing the cost of certain edges and constructing the dual in this altered instance – and this idea can be extracted into a new technique for running the primal-dual schema in the setting of approximation algorithms.

## 1 Introduction

Some of the major open problems left in approximation algorithms are centered around LP-relaxations which researchers believe have not been fully exploited algorithmically, i.e., the best known algorithmic result does not match the best known lower bound on their integrality gaps. One of them is the bidirected cut relaxation for the metric Steiner tree problem [Edm67] and this is the main focus of our paper.

The integrality gap of the bidirected cut relaxation is believed to be very close to 1; the best lower bound known on the gap is  $8/7$ , due to Goemans [Goe96] and Skutella [KPT]. On the other hand, the known upper bound is the same as the weaker undirected cut relaxation which is 2 by a straightforward 2-factor algorithm for it. The only algorithmic results using the bidirected relaxation that we are aware of are: a  $6/5$  factor algorithm for the class of graphs containing at most 3 required vertices [Goe96] and a factor  $3/2$  algorithm for the class of

---

<sup>\*</sup> Work supported by NSF Grant CCF-0728640.

quasi-bipartite graphs, i.e., graphs that do not have edges connecting pairs of Steiner vertices [RV99].

In this paper, we use geometry to develop a new way of lower bounding the cost of the optimal Steiner tree. The best such lower bound can be captured via an LP, which we call the *simplex-embedding LP*. A short description of this LP is that it is an  $l_1$ -embedding of the given metric on a simplex, maximizing a linear objective function. Interestingly enough, the dual of the simplex-embedding LP is a relaxation of the metric Steiner tree problem having the same integrality gap as the bidirected cut relaxation!

A key difficulty faced by researchers in working with the bidirected cut relaxation is the lack of structure in the dual solutions arising mainly due to the asymmetric nature of the primal. We believe our geometric approach would open up new ways to use the primal-dual schema for the bidirected cut relaxation. In particular, we exhibit one dual growing procedure (the EMBED algorithm in Section 2.1) which helps us prove the following property about the bidirected cut relaxation in quasi-bipartite graphs: If the minimum spanning tree is the optimum Steiner tree, then the LP relaxation is exact (Theorem 5).

A second key feature of our paper is a new algorithmic idea of *reduced costs*. We show that modifying the problem instance by reducing costs of certain edges and then running the primal-dual schema allows us to obtain provably better results. We use this idea first to get a simple and fast algorithm which proves an upper bound of  $\sqrt{2}$  on the integrality gap for quasi-bipartite graphs, already improving the previous best of  $3/2$  [RV99]. Using another way of reducing costs, we improve the upper bound to  $4/3$ . This algorithm (similar to the algorithm in [RV99]) doesn't run in strongly polynomial time, unlike the  $\sqrt{2}$  algorithm.

We also give a second geometric LP in which Steiner vertices are not constrained to be on the simplex but are allowed to be embedded "above" the simplex. Again, the dual of this LP is a relaxation of the metric Steiner tree problem. We show that on any instance, the integrality gap of this latter LP is at most that of the bidirected relaxation. It turns out that this LP is in fact exact for Goemans'  $8/7$  example. However, there are examples for which the gap is  $8/7$  even for this relaxation. Could it be that this relaxation has a strictly smaller integrality gap than the bidirected cut relaxation?

## 1.1 Related Work

Historically, the idea of using an extra vertex to get a shorter tree connecting three points on the plane goes back to Torricelli and Fermat in the seventeenth century. The Euclidean Steiner tree problem, in its full generality, was first defined by Gauss in a letter to his student, Schumacher. This problem was made popular by the book of Courant and Robbins [CRS96], who mistakenly attributed it to the nineteenth century geometer, Steiner. The rich combinatorial structure of this problem was explored by many researchers; e.g., see the books by Hwang, Richards and Winter [HRW92] and Ivanov and Tuzhilin [IT94].

The use of the bidirected cut relaxation for the Steiner tree problem goes back to Edmonds [Edm67] who showed the relaxation is exact for the the case of

spanning trees. Wong [Won84] gave a dual ascent algorithm for this relaxation, and Chopra and Rao [CR94a, CR94b] studied the properties of facets of the polytope defined by the relaxation. Goemans and Myung [GM93] study various undirected equivalent relaxations to the bidirected cut relaxation.

The bidirected cut relaxation has interesting structural properties which have been exploited in diverse settings. Let  $\alpha$  denote the integrality gap of this relaxation. [JV02] use this LP for giving a factor 2 budget-balanced group-strategy-proof cost-sharing method for the Steiner tree game; Agarwal and Charikar [AC04] prove that for the problem of multi-casting in undirected networks, the coding gain achievable using network coding is precisely equal to  $\alpha$ . The latter result holds when restricted to quasi-bipartite networks as well. Consequently, for these networks, the previous best bound was  $3/2$  [RV99], and our result improves it to  $4/3$ .

The class of quasi-bipartite graphs is a non-trivial class for the bidirected cut relaxation. In fact, recently Skutella (as reported by Konemann et.al. [KPT]) exhibited a quasi-bipartite graph on 15 vertices for which the integrality gap of the bidirected cut relaxation is  $8/7$ . This ratio matches the erstwhile best known example on general graphs of Goemans [Goe96] where the ratio was met as a limit. Moreover, the best known hardness results for the Steiner tree problem in this class of graphs is quite close to that known in general graphs ( $\frac{128}{127}$  versus  $\frac{96}{95}$ ) [CC02].

The best approximation algorithms for the Steiner tree problem is due to Robbins and Zelikovsky [RZ05]. The authors prove a guarantee of 1.55 for general graphs and also show that when restricted to the quasi-bipartite case, the ratio is 1.28. However, it is not clear if these results would imply an upper bound on the integrality gap of the bidirected cut relaxation. Very recently, Konemann et.al. [KPT] introduced another LP relaxation for the minimum Steiner tree problem which is as strong as the bidirected cut relaxation, and showed that the algorithm of Robbins and Zelikovsky can be interpreted as a primal-dual algorithm on their LP! However, even their interpretation does not prove any upper bound on the integrality gap of their relaxation as they also compare with the optimum Steiner tree and not the optimum LP solution. Nevertheless, they show upper bounds for their relaxation for a larger class of graphs called  $b$ -quasi-bipartite graphs [KPT].

## 1.2 Preliminaries

Let  $G = (V, E)$  be an undirected graph with edge costs  $\mathbf{c} : E \rightarrow \mathbb{R}$ .  $R \subset V$  denotes the set of *required* vertices. The vertices in  $S = V \setminus R$  are called *Steiner* vertices. The Steiner tree problem is to find the minimum cost tree connecting all the required vertices and some subset of Steiner vertices. We abuse notation and denote both the optimum tree and its solution as  $OPT$ . Also, given a set of vertices  $X$ , we denote the minimum spanning tree on  $X$  and its cost as  $MST(X)$ .

---

<sup>1</sup> A graph is  $b$ -quasi-bipartite if on deleting all required vertices, the largest size of any component is at most  $b$ .



The edge costs can be extended to all pairs of vertices such that they satisfy triangle inequality (simply define the cost of  $(uv)$  to be the cost of the shortest path from  $u$  to  $v$ ). This version is called the metric Steiner tree problem. The two versions are equivalent.

Let  $\mathcal{U} := \{U \subsetneq V : U \cap R \neq \emptyset \text{ and } U^c \cap R \neq \emptyset\}$  denote the subsets of  $V$  which contain at least one required vertex but not all. Let  $\delta(U)$  denote the edges with exactly one end point in  $U$ . The undirected cut relaxation of the Steiner tree problem is:

$$\min \left\{ \sum_{e \in E} c(e)x_e : x(\delta(U)) \geq 1, \forall U \in \mathcal{U}; x \geq 0 \right\}$$

The MST on  $R$  is guaranteed to be within factor 2 of the fractional optimum of this LP, so this relaxation has an integrality gap of at most 2. The gap can be arbitrarily close to 2, even for instances of the MST problem.

Now replace each undirected edge  $(uv)$  with two directed arcs  $(\mathbf{uv})$  and  $(\mathbf{vu})$ , each of cost  $c(uv)$ . Call the set of arcs  $\mathbf{E}$ . Fix an arbitrary required vertex  $r$  as root. The set of valid sets  $\mathcal{U}$  are now those which contain the root but not all the required vertices. If the edges of a Steiner tree are directed to point away from the root, then at least one edge is in the cut set  $\delta^+(U)$  of arcs going out of  $U$ . This gives the bidirected cut relaxation.

$$\min \left\{ \sum_{e \in \mathbf{E}} c(e)x_e : x(\delta^+(U)) \geq 1, \forall U \in \mathcal{U}; x \geq 0 \right\} \tag{1}$$

We denote the optimum of the above LP on a graph  $G$  as  $BCR(G)$ . A graph is called *quasi-bipartite* if there are no edges between two Steiner vertices. The Steiner tree problem is NP-Hard even when restricted to the class of quasi-bipartite graphs [BP89, CC02].

**Organization:** In Section 2 we show the geometric theorem giving the lower bound, and other results relevant to it. In Sections 3 and 4 we give our  $\sqrt{2}$  and  $\frac{4}{3}$  factor approximation algorithms respectively.

## 2 A Geometric Lower Bound and Its Consequences

We first present a special case of the geometric theorem, for the sake of ease of presentation. Let  $\Delta_k$  be the unit simplex in  $\mathbb{R}^k$ , that is,  $\Delta_k := \{x \in \mathbb{R}^k : \sum_{i \in [k]} x(i) = 1\}$ , where  $x(i)$  is the  $i$ th coordinate of  $x$ . The corners of  $\Delta_k$  are the unit vectors in  $\mathbb{R}^k$ . Let  $T$  be any Steiner tree in  $\Delta_k$  connecting the corners, that is,  $T$  is a tree whose vertices are the corners and any number of points in  $\Delta_k$ . Define the distance between two points to be half the  $l_1$ -distance, also called the *variational distance*; for any two points  $x, y \in \Delta_k$ ,  $d(x, y) := \frac{1}{2} \sum_{i=1}^k |x(i) - y(i)|$ . (The half is so that two corners are at a distance of 1). Let  $d(T) := \sum_{e \in T} d(e)$ . Then

**Theorem 1.**  $d(T) \geq k - 1$ .

Note that if  $T$  was a spanning tree, then the relation holds with equality since any two corners are at a distance of 1. The theorem says that Steiner points don't improve upon the MST, w.r.t  $d()$ . This is somewhat counter-intuitive, since in most geometric spaces, the Steiner points do improve upon the MST. What is special here is the  $l_1$ -distance, and the location of the points on the simplex.

*Proof.* Let  $R = \{e_1, e_2, \dots, e_k\}$  be the unit vectors in  $\mathbb{R}^k$ . The proof follows by a careful counting argument. First of all, we get rid of the absolute values that occur in the  $l_1$  distance. For every edge  $(x, y)$  in  $T$ , and  $i \in [k]$ , if  $x$  is on the  $T$ -path from  $y$  to  $e_i$ , then we lower bound  $|x(i) - y(i)|$  by  $x(i) - y(i)$  (and vice versa). This way the sum  $\sum_{e \in T} d(e)$  is lower bounded by a linear combination of the  $x(i)$ 's, in which the coefficient of  $x(i)$  is  $\frac{1}{2}(\text{deg}_T(x) - 2)$ . This is because  $x(i)$  occurs with a positive sign for each edge incident at  $x$  except one, the first edge on the  $T$ -path from  $x$  to  $e_i$ . The exception is  $e_i(i)$  which always occurs with a positive sign, and hence has a coefficient of  $\frac{1}{2}\text{deg}_T(e_i)$ . Therefore,

$$\begin{aligned} \sum_{e \in T} d(e) &\geq \frac{1}{2} \sum_{x \in T, i \in [k]} (\text{deg}_T(x) - 2) x(i) + \frac{1}{2} \sum_{i \in [k]} 2e_i(i) \\ &= \frac{1}{2} \sum_{x \in T} (\text{deg}_T(x) - 2) + \sum_{i \in [k]} 1 \\ &= k - 1. \end{aligned}$$

where the equality in the second line holds because  $\sum_{i \in [k]} x(i) = 1$  and the last equality follows from the fact that in a tree  $\sum_{x \in T} \text{deg}(x) = 2|T| - 2$ .  $\square$

The general theorem allows two concessions on the location of the points: first, the points need not be on the unit simplex, all point are in the  $\lambda$ -simplex  $\Delta_k^{(\lambda)}$ , defined as  $\{x \in \mathbb{R}^k : \sum_{i \in [k]} x(i) = \lambda\}$ , for some parameter  $\lambda > 0$ . The second is that the required points need not be at the corners of the simplex. In particular, we consider any embedding  $z$  that maps every vertex  $u \in V = R \cup S$  to a point  $z_u \in \Delta_k^{(\lambda)}$ , where  $k = |R|$ . Identify the required vertices with the dimensions, and define  $\gamma(z) := \sum_{i \in [k]} z_i(i) - \lambda$ . Note that when the required vertices are at the corners of a unit simplex,  $\gamma(z) = k - 1$ . As before, let  $T$  be any Steiner tree connecting all points in  $R$ ,  $d(u, v) = \frac{1}{2} \sum_{i=1}^k |z_u(i) - z_v(i)|$  and  $d(T) = \sum_{e \in T} d(e)$ . The above proof can be easily be modified to give:

**Theorem 2.**  $d(T) \geq \gamma(z)$ .

This theorem can be used to get a lower bound on the minimum Steiner tree as follows: Given a graph, a *valid* embedding of the vertices onto any  $\lambda$ -simplex is such that for all edges  $e$ ,  $\mathbf{c}(e) \geq d(e)$ . Now given any valid embedding  $z$ , for any tree  $T$ ,  $\mathbf{c}(T) \geq d(T) \geq \gamma(z)$ . In particular, we have

---

<sup>2</sup> An easy counting argument shows that  $d(T) \geq \frac{1}{2}(k - 1)$ .

**Theorem 3.** *If  $z$  is a valid embedding then*

$$OPT \geq \gamma(z).$$

Since the above holds for any embedding, the best lower bound is given by  $\max \{\gamma(z) : z \text{ is valid}\}$ . Quite interestingly, this maximum value for any graph is equal to  $BCR(G)$ .

**Theorem 4.** *Given any graph  $G$ ,*

$$\max \{\gamma(z) : z \text{ valid embedding of } G\} = BCR(G).$$

It is not too hard to see that the maximum in the above theorem can be obtained via a polynomial sized linear program, which we call the simplex-embedding LP. In fact, the dual to this program turns out to be a relaxation for the Steiner tree problem and the proof of the above theorem follows by showing its equivalence with the bidirected cut relaxation. In [GM93], Goemans and Myung provide two vertex weighted relaxations which are equivalent to the bidirected cut relaxation. Although our relaxation is different, our proof of equivalence follows on similar lines.

*Proof.* The simplex-embedding LP is as follows:

$$\begin{aligned} & \text{maximize } \gamma(z) = \sum_{i \in [k]} z_i(i) - \lambda & (2) \\ & \text{subject to } \sum_{i \in [k]} z_v(i) = \lambda, \quad \forall v \in V \\ & \quad z_v(i) - z_u(i) \leq d_i(uv), \quad \forall i \in [k], (uv) \in E \\ & \quad z_u(i) - z_v(i) \leq d_i(uv), \quad \forall i \in [k], (uv) \in E \\ & \quad \frac{1}{2} \sum_{i \in [k]} d_i(uv) \leq c(uv), \quad \forall (uv) \in E \\ & \quad z_v(i), d_i(uv) \geq 0, \quad \forall v \in V, i \in [k], (uv) \in E \end{aligned}$$

Taking duals, we get the following LP (after a scaling step)

$$\begin{aligned} & \text{minimize } \sum_{e \in E} c(e)x_e & (3) \\ & \text{subject to } x_{uv} \geq f_i(uv) + f_i(vu), \quad \forall i \in [k], (uv) \in E \\ & \quad \sum_{v:(uv) \in E} (f_i(uv) - f_i(vu)) \geq \alpha(v), \quad \forall i \in [k], v \in V - i \\ & \quad \sum_{v:(iv) \in E} (f_i(iv) - f_i(vi)) \geq \alpha(i) + 2, \quad \forall i \in [k] \\ & \quad \sum_v \alpha(v) = -2 \\ & \quad f_i(uv), f_i(vu), x_{uv} \geq 0, \quad \forall (uv) \in E, i \in [k] \end{aligned}$$

To interpret LP [3](#), one can think of  $x_e$  as capacity of edge  $e$ . There are  $k$  flows -  $f_i$  for every required vertex  $i$  each satisfying the capacity constraint and moreover, the supplies (excess flows) for  $f_i$  at every vertex  $v$  is  $\alpha(v)$  (it could be negative) except at the vertex  $i$ , where it is  $\alpha(i) + 2$ . The last equality constraint implies the total supplies sum up to zero.

Theorem [4](#) follows by showing there exists a feasible solution to LP [1](#) of value  $\Gamma$  if and only if there exists a feasible solution to LP [3](#) of value  $\Gamma$ . We only give a sketch for brevity and defer the details to a full version [\[CDV07\]](#).

**LP [1](#)  $\rightarrow$  LP [3](#):** Let  $\{y_{(uv)}\}_{(uv) \in E}$  be a feasible solution of LP [1](#) of cost  $\Gamma$  with root  $r$ . The corresponding solution to LP [3](#) is as follows:  $x_{uv} := y_{(uv)} + y_{(vu)}$  and  $\alpha(v)$  is the *supply* at vertex  $v$  which is the difference of the outgoing  $y_{(vu)}$ 's and the incoming  $y_{(uv)}$ 's, except at  $r$  where  $\alpha(r)$  is the supply  $-2$ . What remains is to describe the flows. The flow corresponding to  $r$  just mimics the  $y_{(uv)}$ 's. It is easy to see every constraint is till now satisfied. To get the flows corresponding to another required vertex  $j$ , we use the fact that the *minimum*  $r - j$  cut in the digraph with arc set  $E$  and capacities  $y_{(uv)}$ 's is at least 1 since the solution is feasible for LP [1](#). This implies there is a standard flow  $g_{rj}$  from  $r$  to  $j$  in this di-graph. The flow  $f_j$  is found by *subtracting*  $2g_{rj}$  from  $f_r$ . Note that this changes the supplies of no vertex other than  $r$  and  $j$ , and for these, it changes as it should change. Thus the solution is feasible for LP [3](#) and is of value  $\Gamma$ .

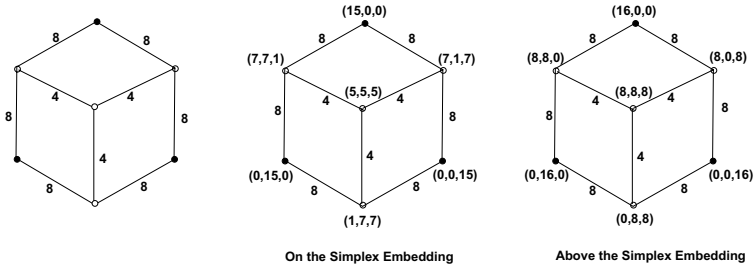
**LP [3](#)  $\rightarrow$  LP [1](#):** Let  $(\{x\}, \{f_i\}, \{\alpha\})$  (respectively over edges, arcs and vertices) be a solution to LP [3](#). WLOG by adding circulations if necessary, we can assume for all edges  $(uv)$ , and  $i$  we have  $x_{uv} = f_i((uv)) + f_i((vu))$ . For LP [1](#), let  $r$  be the chosen root. Then the solution is:  $y_{(uv)} := f_r((uv))$ . To see feasibility for LP [1](#), we must show across any cut  $S$  separating  $r$  and a required vertex  $j$ , we have  $\sum_{(uv) \in \delta^+(S)} f_r((uv)) \geq 1$ . To see this, consider the flow  $g_{rj}$ : the difference between  $f_r$  and  $f_j$ . To be precise:

$$g_{rj}((uv)) = \max[0, f_r((uv)) - f_j((uv))] + \max[0, f_j((vu)) - f_r((vu))]$$

It is not hard to see for any arc  $(uv)$ ,  $g_{rj}(uv) \leq 2f_r((uv))$ . The proof sketch ends by noting  $g_{rj}$  is a standard flow from  $r$  to  $j$  of value 2, implying across any cut  $S$  as above, 2 units (and hence at least 1 unit of  $f_r$ ) of  $g_{rj}$  would go across.  $\square$

Interestingly, the above theorems hold even when the Steiner vertices are embedded ‘‘above’’ the simplex. That is, if we have for all Steiner vertices  $v$ ,  $\sum_{i \in [k]} z_v(k) \geq \lambda$  rather than equality. Maximizing over these embeddings give us a better lower bound on OPT and in fact, as we see in Figure [1](#), sometimes it is strictly better. Thus, we obtain an LP relaxation which could be *tighter* than the bidirected cut relaxation. Although the remainder of the paper does not concern this relaxation, it is an intriguing question if the integrality gap of this revised LP is strictly smaller than that of the bidirected cut relaxation.

**Remark:** We should remark that the idea of embedding vertices of a graph onto a simplex is not new. Calinescu et.al. [\[CKR98\]](#) use a similar LP to obtain



**Fig. 1.** Integrality gap of the bidirected cut relaxation for the graph is known to be  $16/15$  (due to Goemans). The middle figure shows an embedding on the simplex attaining a value of 15. The figure to the right shows how we can get a higher value if we allow Steiner vertices to move above the simplex. Note that the Steiner vertex at the center is not on the 16-simplex.

approximation algorithms for the multi-way cut problem. However, a key difference is that there is a primal relaxation while ours is dual. It is not clear if a certain duality between the two LP's can be established.

### 2.1 An Embedding Algorithm

In this section, we describe a dual growing procedure EMBED which given a quasi-bipartite graph  $G$  and a cost function  $\mathbf{c}$  does the following.

**Case 1:** If  $MST(R)$  is the optimal Steiner tree, then it returns a feasible embedding  $z$  such that  $\gamma(z) = MST(R)$ . Note, in this case,  $MST(R) = OPT = BCR$ , since  $MST(R) \geq OPT \geq BCR \geq \gamma(z)$ .

**Case 2:** Or, returns a Steiner vertex  $v$  whose addition *strictly* helps the MST on  $R$ , that is,  $MST(R \cup v) < MST(R)$ . We say that EMBED *crystallizes*  $v$ .

The following theorem is immediate.

**Theorem 5.** *Given a quasi-bipartite instance  $G$ , if the addition of no Steiner vertex reduces the cost of  $MST(R)$ , then  $MST(R) = BCR$ . In particular the integrality gap for the instance is 1.*

Note that the theorem implies the following important property about the bidirected cut relaxation for quasi-bipartite graphs: If the minimum spanning tree is optimal, then the relaxation is exact. Only the above property of EMBED is used beyond this section. The rest of the section describes EMBED.

**Notation:** *Given an embedding  $z : V \rightarrow \Delta_k^{(\lambda)}$  and the distance  $d(\cdot, \cdot)$  induced by it, call an edge  $(u, v)$  tight if  $d(u, v) = \mathbf{c}(uv)$ . Call  $(u, v)$  under-tight or over-tight if the distance is strictly smaller or larger, respectively.*

The following is a continuous description of the algorithm, which can be easily discretized. The algorithm has a notion of time. It starts at time  $t = 0$  and time increases at unit rate. At any time  $t$ , all required vertices are on the  $t$ -simplex, all Steiner vertices are below the  $t$ -simplex, and no edge is over-tight. The algorithm

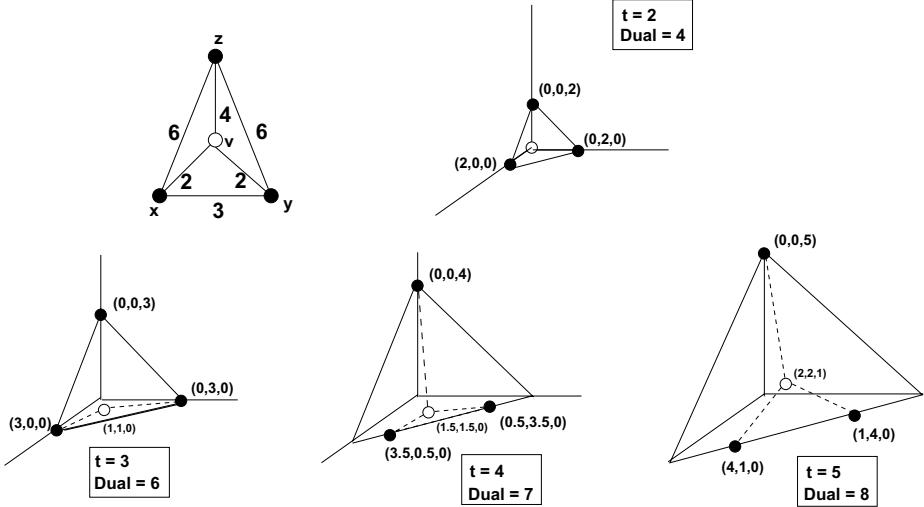
maintains a set of tight terminal-terminal edges  $T$ , which form a forest at any time  $t$ . Let  $K$  denote a connected component of required vertices formed with the edges of  $T$ . At time  $t = 0$ , the algorithm starts with  $T = \emptyset$  and the components are singleton required vertices. All vertices start at the origin at  $t = 0$ .

**Required vertices:** For each component  $K$  and required vertex  $i \in K$ , the algorithm increases the  $j$ th coordinate of  $i$  at rate  $1/|K|$ , for each  $j \in K$ . Clearly, this will keep required vertex  $i$  on the  $t$ -simplex. When an edge  $(ij)$  goes tight, the algorithm merges the components containing  $i$  and  $j$  and adds  $(ij)$  to  $T$ . It is instructive to note that when restricted to only required vertices, this actually mimics Kruskal’s MST algorithm.

**Steiner vertices:** A Steiner vertex  $v$  remains at the origin until it *links* to a required vertex. It links to required vertex  $i$  at time  $t = c(iv)$ , if it is not already linked to another required vertex in the same component as  $i$ . The edge  $(iv)$  is called a *link*. We say that  $v$  is linked to a component  $K$  if it is linked to any required vertex in  $K$ . For each component  $K$  that  $v$  is linked to, the coordinates of  $v$  corresponding to  $K$  increase at rate  $1/|K|$ .

The algorithm terminates if the number of components becomes 1 (Case 1) or a Steiner vertex  $v$  hits the simplex (Case 2). The example in Figure 2 illustrates the algorithm on a graph with three required vertices.

We now show the above procedure satisfies the conditions. In Case 1, the algorithm returns the embedding obtained after running the following projection



**Fig. 2.** Snapshots of the running of EMBED on the graph above at times  $t = 2, 3, 4, 5$ . At time  $t = 2$ , the Steiner vertex  $v$  links to the required vertices  $x$  and  $y$ , and increases its  $x$  and  $y$  coordinates at rate 1. At time  $t = 3$ ,  $x, y$  merge. The edge  $(x, y)$  goes into *Remove*( $v$ ). At time  $t = 4$ ,  $v$  links to  $z$ , and moves in the  $z$ th coordinate as well. At  $t = 5$ , it hits the 5-simplex, terminating the algorithm. The tree shown with dotted lines pays exactly for the dual and is cheaper than the MST.

step. If Case 1 happens at time  $t = \lambda$ , then the algorithm projects each Steiner vertex onto the  $\lambda$ -simplex. For every Steiner vertex  $v$  and coordinate  $j$ ,  $z_v(j) \leftarrow z_v(j) \frac{\lambda}{\|z_v\|_1}$ . The coordinates of the required vertices are kept the same. It is easy to show that  $z$  is feasible, that is, no edge is over-tight. We need to show that tree  $T$  has cost  $\gamma(z)$ . In fact we prove something stronger. Given any connected component  $K$ , denote the restriction of  $T$  to  $K$  as  $T[K]$ .

**Lemma 1.** *At any instant of time  $t$ , for any connected component  $K$ ,*

$$c(T[K]) = \sum_{i \in K} z_i(i) - t.$$

*Proof.* At time  $t = 0$ , the lemma holds vacuously. Since the quantity  $\sum_{i \in K} z_i(i)$  increases at the same rate as time, we need to prove the lemma only in the time instants when two components merge. Suppose  $K, K'$  merge at time instant  $t$  due to edge  $(ij)$  which comes in the tree, with  $i \in K, j \in K'$ . Note  $d(i, j) = c(ij) = t$ . So for the new connected component  $K \cup K'$ ,  $\sum_{i \in K \cup K'} z_i(i) - t = \sum_{i \in K} z_i(i) - t + \sum_{i \in K'} z_i(i) - t + t = c(T[K]) + c(T[K']) + c(ij) = c(T[K \cup K'])$ .  $\square$

In Case 2, when  $v$  hits the simplex, the algorithm returns  $v$  as the Steiner vertex helping the minimum spanning tree. In fact, we show that if  $v$  is linked to  $K_1, \dots, K_r$  when it hits the simplex, then  $v$  helps the MST of the required vertices in  $P = \bigcup_l K_l$ . This suffices since  $\bigcup_l T[K_l]$  can be extended to an MST of  $R$ .

With each Steiner vertex  $v$ , we associate a subset of edges  $Remove(v)$  of  $T$ . Suppose  $v$  is linked to  $K$  and  $K'$  and these merge at time  $t$ , due to edge  $(ij)$ ,  $i \in K$  and  $j \in K'$ . At this point,  $(ij)$  is added to the set  $Remove(v)$ . Thus, a Steiner vertex may have more than one link into the same component, but for each extra link, there is an edge in  $Remove(v)$ . Let  $T_v$  be the tree formed by adding all the links incident at  $v$  to  $\bigcup_l T[K_l]$  and deleting  $Remove(v)$ . The proof of the following lemma is very similar to that of Lemma 1.

**Lemma 2.** *At any instant of time,  $c(T_v) = \sum_{i \in P} z_i(i) - \sum_{i \in P} z_v(i)$ .*

Hence when  $v$  hits the simplex,  $\sum_{i \in P} z_v(i) = t$ , and so  $c(T_v) = \sum_{i \in P} z_i(i) - t < MST(P)$ .

**Remark:** Note that the above algorithm and analysis do not use the fact the cost satisfies triangle inequality. We would need this for our algorithms to work.

### 3 The $\sqrt{2}$ Factor Approximation Algorithm

**Notation 1.**  $MST(U; \mathbf{c})$  denotes the minimum cost spanning tree on vertices  $U$  given the costs  $\mathbf{c}$ . Based on the context, it also denotes the cost of this tree.

We start by giving a high level idea of our algorithm. The algorithm will return a cost  $\mathbf{c}_2$  and a subset of Steiner vertices  $X \subseteq S$  such that

1. The optimal Steiner tree w.r.t.  $\mathbf{c}_2$  is the MST. Equivalently, EMBED when run on  $G, \mathbf{c}_2$  terminates with a feasible embedding  $z$  with  $\gamma(z) = MST(R; \mathbf{c}_2)$ .
2.  $MST(X \cup R; \mathbf{c}) \leq \sqrt{2} \cdot MST(R; \mathbf{c}_2)$

The costs  $\mathbf{c}_2$  will be only smaller than  $\mathbf{c}$ ; therefore,  $z$  is also feasible for  $\mathbf{c}$ . Hence, the two conditions imply that we get a factor  $\sqrt{2}$  approximation.

Initially,  $X = \emptyset$  and we obtain  $\mathbf{c}_2$  by reducing the costs of the required-required edges by a factor of  $\sqrt{2}$  and leaving the costs of required-Steiner edges unchanged. We denote the reduced cost at this point as  $\mathbf{c}_1$  which we use later. Clearly Condition 2 is satisfied now, and will remain an *invariant* of the algorithm.

Suppose that condition 1 is not satisfied, that is, when EMBED is run on  $G, \mathbf{c}_2$ , a Steiner vertex  $v \in S$  is crystallized. At this point, the algorithm adds  $v$  to  $X$ , and will modify  $\mathbf{c}_2$  by reducing the costs of certain required-required edges further, as detailed below. This has the effect that if EMBED is run with these new costs,  $v$  does not crystallize, while still maintaining the invariant. Hence in each iteration, a new Steiner vertex is added to  $X$ , implying termination in at most  $|S|$  rounds.

We now give the intuition behind modifying the costs so that the invariant is maintained. The first step of scaling all the required-required edges acts as a “global filter” which filters out Steiner vertices that only help a little. If a Steiner vertex  $v$  does crystallize, then adding it to  $X$  reduces the cost of  $MST(R \cup X; \mathbf{c})$  so much that decreasing the cost of required-required edges “local” to it to  $\frac{1}{2}$  of the original costs still maintains the invariant. This requires an involved argument (Theorem 6) that amortizes the improvements due to all the vertices previously added to  $X$ . This has the additional required effect that  $v$  itself is filtered out.

Now the formal description of the algorithm follows.

**Definition 1.** Applying the global filter with parameter  $\rho > 1$  gives a cost  $\mathbf{c}_1$  defined as  $\mathbf{c}_1(ij) = 1 \frac{\mathbf{c}(ij)}{\rho}$  for all  $i, j \in R$ , and  $\mathbf{c}_1(iv) = \mathbf{c}(iv)$  for all  $i \in R$  and  $v \in S$ .

**Definition 2.** Applying a local filter w.r.t  $X \subseteq S$  gives a cost  $\mathbf{c}_2$ . Let  $T_1 = MST(R \cup X; \mathbf{c}_1)$ , and for each  $u \in X$ ,  $Clos(u)$  denote the closest required vertex to  $u$ . The cost  $\mathbf{c}_2$  after applying the local filter w.r.t  $X$  is defined as  $\mathbf{c}_2(Clos(u), j) = \frac{1}{2}\mathbf{c}(Clos(u), j)$  (half the original cost), for every  $u \in X$  and  $j \in R$  ( $j \neq Clos(u)$ ) that is adjacent to  $u$  in  $T_1$ .  $\mathbf{c}_2(e) = \mathbf{c}_1(e)$  (the global filter is retained) otherwise.

Algorithm PRIMAL-DUAL

1. Apply global filter with parameter  $\rho = \sqrt{2}$  to get  $\mathbf{c}_1$ .  
Initialize  $X \leftarrow \emptyset$ ;  $\mathbf{c}_2 \leftarrow \mathbf{c}_1$ .
2. **Repeat** till EMBED returns  $z$   
Run EMBED on  $G, \mathbf{c}_2$ .  
If EMBED returns  $v$  then  
 $X = X \cup v$ ; Apply local filter w.r.t  $X$  to get  $\mathbf{c}_2$ .
3. Return  $T_1 = MST(R \cup X; \mathbf{c}_1), z$ .



**Theorem 6.** *The algorithm PRIMAL-DUAL terminates in at most  $|S|$  rounds, returning a Steiner tree  $T_1$  and a feasible embedding  $z$  of  $G, \mathbf{c}$  such that  $\mathbf{c}(T_1) \leq \sqrt{2} \cdot \gamma(z) \leq \sqrt{2} \cdot OPT$ .*

*Proof (Sketch).* Let  $T_1 = E_0 \cup E_1$ , where  $E_0$  denotes the required-Steiner edges and  $E_1$  denotes the required-required edges of  $T_1$ . We bound the costs of these two sets separately. Let  $E_2$  be the set of edges modified by the local filter, that is,  $e$  such that  $\mathbf{c}_2(e) = \frac{1}{2}\mathbf{c}(e)$ . Define  $T_2$  to be  $E_2 \cup E_1$ . It can be shown that  $T_2$  is an MST with costs  $\mathbf{c}_2$ , and hence  $\mathbf{c}_2(T_2) = \gamma(z)$ . We have  $\mathbf{c}(T_1) = \mathbf{c}(E_0) + \mathbf{c}(E_1)$ ,  $\mathbf{c}_2(T_2) = \mathbf{c}_2(E_2) + \mathbf{c}_2(E_1)$  and it is enough to prove that

$$- \mathbf{c}(E_0) \leq \sqrt{2}\mathbf{c}_2(E_2).$$

This is essentially a consequence of the observation that  $\mathbf{c}_1(T_1) \leq MST(R; \mathbf{c}_1)$ . Since  $T_2 = E_1 \cup E_2$  is also a spanning tree of  $R$ , we get  $\mathbf{c}_1(T_1) \leq \mathbf{c}_1(T_2)$ . Expanding the costs, we get

$$\mathbf{c}_1(E_0) + \mathbf{c}_1(E_1) \leq \mathbf{c}_1(E_2) + \mathbf{c}_1(E_1).$$

Since  $E_0$  are required vertex-Steiner edges,  $\mathbf{c}_1(E_0) = \mathbf{c}(E_0)$ .  $\mathbf{c}_1(E_2) = \mathbf{c}(E_2)/\sqrt{2} = \sqrt{2}\mathbf{c}_2(E_2)$  by definition, giving us  $\mathbf{c}(E_0) \leq \sqrt{2}\mathbf{c}_2(E_2)$ .

$$- \mathbf{c}(E_1) \leq \sqrt{2}\mathbf{c}_2(E_1).$$

Since  $E_1$  costs are not modified by the local filter,  $\mathbf{c}_2(E_1) = \mathbf{c}_1(E_1)$  and in fact the relation holds with equality.  $\square$

In fact, the above algorithm has a faster implementation. Although the algorithm constructs the set  $X$  in a certain order, it turns out that the order does not matter. Hence it is enough to simply apply the global filter and go through the Steiner vertices (in any order) once, picking the ones that help.

**Algorithm REDUCED ONE-PASS HEURISTIC**

1. Apply global filter with parameter  $\rho = \sqrt{2}$  to get  $\mathbf{c}_1$ .  
Initialize  $X \leftarrow \emptyset$ ;
2. For all  $v \in S$ ,  
If  $MST(R \cup X \cup v; \mathbf{c}_1) < MST(R \cup X; \mathbf{c}_1)$ , then  
 $X = X \cup v$ ;
3. Return  $T_1 = MST(R \cup X; \mathbf{c}_1)$ .

**Theorem 7.** *There exists a feasible embedding  $z$  of  $G, \mathbf{c}$  such that for  $T_1$  returned by Algorithm REDUCED ONE-PASS HEURISTIC,  $\mathbf{c}(T_1) \leq \sqrt{2} \cdot \gamma(z)$ .*

The proof of Theorem 7 is similar to Theorem 6. Note that the above algorithm makes at most  $|S|$  minimum spanning tree computations and is hence is very efficient. In particular, it runs in strongly polynomial time.

## 4 The $\frac{4}{3}$ Factor Approximation Algorithm

The primal-dual  $\frac{4}{3}$  approximation algorithm is along the lines of the one in the previous section, with the major difference being that it drops Steiner vertices from  $X$  when beneficial. The other differences are that it applies the global filter with  $\rho = 4/3$ , and the definition of a local filter is somewhat different. And like the earlier algorithm, the order of vertices picked/dropped does not matter. As a result it can be implemented as a local search algorithm with an extra global filtering step, which is what we present here.

Algorithm REDUCED-LOCAL-SEARCH

1. Apply global filter with parameter  $\rho = 4/3$  to get  $\mathbf{c}_1$ .  
Initialize  $X \leftarrow \emptyset, T_1 = MST(R; \mathbf{c}_1)$ ;
2. Repeat
  - If  $\exists v$  such that  $MST(R \cup X \cup v; \mathbf{c}_1) < \mathbf{c}_1(T_1), X = X \cup v$ .
  - If  $\exists v$  such that  $MST(R \cup X \setminus v; \mathbf{c}_1) < \mathbf{c}_1(T_1), X = X \setminus v$ .
  - $T_1 = MST(R \cup X; \mathbf{c}_1)$ .
 Until No such  $v$  exists.
3. Return  $T_1$ .

The plain local search algorithm (without the global filtering step) was studied [RV99] who showed that this algorithm gives a  $3/2$  factor approximation for quasi-bipartite graphs. This factor is tight. So the simple modification of applying a global filter provably improves the performance of this algorithm. It was shown in [Riz03] that this algorithm can be implemented efficiently.

We show that  $T_1$  returned by the algorithm is within  $4/3$  of the optimal by exhibiting an embedding  $z$  of value greater than  $3/4$  times the cost of  $T_1$ . As in Section 3, the analysis proceeds by defining cost  $\mathbf{c}_2$  and constructing tree  $T_2$ . The factor  $4/3$  comes from the parameter  $\rho$  used in the global filter and the following property of  $T_1$ .

**Lemma 3.** *The degree of every Steiner vertex in  $T_1$  is at least 4.*

*Proof.* It is easy to see that  $T_1$  doesn't have vertices of degree 1 or 2. Suppose there existed a Steiner vertex  $v \in T_1$  with  $deg(v) = 3$ . Let  $a, b, c$  be the required vertices connected to  $v$  and assume  $\mathbf{c}_1(va) \leq \mathbf{c}_1(vb) \leq \mathbf{c}_1(vc)$  without loss of generality. Now by triangle inequality property of  $\mathbf{c}$ , we know  $\mathbf{c}(va) + \mathbf{c}(vb) \geq \mathbf{c}(ab)$ . Since  $\mathbf{c}(va) = \mathbf{c}_1(va)$  and  $\mathbf{c}(vb) = \mathbf{c}_1(vb)$ , we get  $\frac{3}{4}(\mathbf{c}_1(va) + \mathbf{c}_1(vb)) \geq \frac{3}{4}\mathbf{c}(ab) = \mathbf{c}_1(ab)$ . Similarly  $\frac{3}{4}(\mathbf{c}_1(va) + \mathbf{c}_1(vc)) \geq \mathbf{c}_1(ac)$ . Thus  $\mathbf{c}_1(ab) + \mathbf{c}_1(ac) \leq \frac{3}{4}(2\mathbf{c}_1(va) + \mathbf{c}_1(vb) + \mathbf{c}_1(vc)) \leq \mathbf{c}_1(va) + \mathbf{c}_1(vb) + \mathbf{c}_1(vc)$ . Thus  $MST(R \cup X)$  would choose  $(ab)$  and  $(ac)$ , rather than choosing  $(va), (vb), (vc)$ .  $\square$

**Theorem 8.** *For the tree  $T_1$  returned by REDUCED-LOCAL-SEARCH, there exists a feasible embedding  $z$  such that  $\mathbf{c}(T_1) \leq \frac{4}{3} \cdot \gamma(z)$ .*

*Proof (Sketch).* As in the proof of Theorem 6, denote the edges of  $T_1$  as  $E_1 \cup E_0$ . Define  $\mathbf{c}_2$  as: For every Steiner vertex  $v \in T_1$  and for every  $j \neq \text{Clos}(v)$  connected to  $v$  in  $T_1$ , let  $\mathbf{c}_2(\text{Clos}(v), j) = \mathbf{c}_1(vj)$ . Note that  $\mathbf{c}_1(vj) \leq \mathbf{c}_1(\text{Clos}(v), j)$ , for otherwise  $T_1$  would have picked  $(\text{Clos}(v), j)$  instead of  $(vj)$ . Call these required vertex-required vertex edges *diminished*. For every other edge,  $\mathbf{c}_2(e) := \mathbf{c}_1(e)$ . Let  $E_2$  be the set of diminished edges and let  $T_2 := E_1 \cup E_2$ , be a required vertex spanning tree. By the conditions of the algorithm, since  $T_1$  is an MST of  $R \cup X$  with costs  $\mathbf{c}_1$  and no Steiner vertices help  $X$ ,  $T_2$  is an MST of  $R$  with costs  $\mathbf{c}_2$  and no Steiner vertex helps  $T_2$ . Thus, by Theorem 5 running EMBED on  $G$ ,  $\mathbf{c}_2$  returns a feasible embedding  $z$  of value  $\mathbf{c}_2(T_2)$ . We now bound the cost of  $T_1$ .

We have  $\mathbf{c}(T_1) = \mathbf{c}(E_1) + \mathbf{c}(E_0) = \mathbf{c}(E_1) + \mathbf{c}_1(E_0)$ . Note that  $\mathbf{c}_2(T_2) = \mathbf{c}_1(E_1) + \mathbf{c}_2(E_2)$  since  $E_1$  is not diminished. As in the proof of Theorem 6, we argue term by term. By definition we have  $\mathbf{c}(E_1) = \frac{3}{4}\mathbf{c}_1(E_1)$ .

Every Steiner vertex  $v \in T_1$  contributes  $\deg(v) - 1$  edges to  $E_2$  and  $\deg(v)$  edges in  $E_0$ , where  $\deg(v)$  is the degree of  $v$  in  $T_1$ . By definition the  $\deg(v) - 1$  edges have cost exactly the cost of the largest  $\deg(v) - 1$  edges of the  $\deg(v)$  edges it contributes to  $E_0$ . By lemma 3,  $\deg(v) \geq 4$  and thus we get  $\mathbf{c}_1(E_0) \leq \frac{3}{4}\mathbf{c}_2(E_2)$ . Adding, we get  $\mathbf{c}(T_1) \leq \frac{4}{3}\mathbf{c}_2(T_2) = \frac{4}{3}\gamma(z)$ .  $\square$

## 5 Discussion

Clearly the most important question to address is whether the geometric approached to the bidirected cut relaxation describe here can be extended to general graphs. In fact, there is a natural generalization of the EMBED procedure described above to the case where there are Steiner-Steiner edges; however, it has not yielded any results for the general case. As noted above, one crucial property possessed by quasi-bipartite graphs is Theorem 5: if the spanning tree is optimal, then the relaxation is exact. However, this property is not satisfied by general graphs. An interesting question would be upper bounding the gap in such instances, and then perhaps our techniques of reducing costs may be useful.

## References

- [AC04] Agarwal, A., Charikar, M.: On the advantage of network coding for improving network throughput. In: Proceedings of the IEEE Information Theory Workshop (2004)
- [BP89] Bern, M., Plassman, P.: The Steiner problem with edge lengths 1 and 2. Inform. Process. Lett. 32, 171–176 (1989)
- [CC02] Chlebik, M., Chlebikova, J.: Approximation hardness of the steiner tree problem on graphs. In: Proceedings of Scandinavian Workshop on Algorithm Theory (2002)
- [CDV07] Chakrabarty, D., Devanur, N.R., Vazirani, V.V.: New geometry-inspired relaxations and algorithms for the Metric Steiner Tree Problem (2007), <http://www.cc.gatech.edu/~deepc>
- [CKR98] Calinescu, G., Karloff, H., Rabani, Y.: An improved approximation algorithm for multiway cut. In: STOC (1998)

- [CR94a] Chopra, S., Rao, M.R.: The Steiner tree problem I: Formulations, compositions and extension of facets. *Math. Programming* 64, 209–229 (1994)
- [CR94b] Chopra, S., Rao, M.R.: The Steiner tree problem II: Properties and classes of facets. *Math. Programming* 64, 231–246 (1994)
- [CRS96] Courant, R., Robbins, H., Stewart, I.: *What Is Mathematics?: An Elementary Approach to Ideas and Methods*. Oxford Paperbacks (1996)
- [Edm67] Edmonds, J.: Optimum branchings. *Journal of Research of the National Bureau of Standards. Section B* 71, 233–240 (1967)
- [GM93] Goemans, M., Myung, Y.: A catalog of Steiner tree formulations. *NETWORKS* 23, 19–23 (1993)
- [Goe96] Goemans, M.: Personal communication (1996)
- [HRW92] Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem. *Annals of Discrete Mathematics*, vol. 53. North-Holland, Amsterdam, Netherlands (1992)
- [IT94] Ivanov, A.O., Tuzhilin, A.A.: *The Steiner problem and its generalizations*. CRC Press, BocaRaton, Ann Arbor, London, Tokyo (1994)
- [JV02] Jain, K., Vazirani, V.V.: Equitable cost allocations via primal-dual-type algorithms. In: *Proceedings of 33rd ACM Symposium on Theory of Computing* (2002)
- [KPT] Konemann, J., Pritchard, D., Tan, K.: A partition based relaxation for Steiner trees (manuscript)
- [Riz03] Rizzi, R.: On Rajagopalan and Vazirani’s  $3/2$ -approximation bound for the iterated 1-Steiner heuristic. *Information Processing Letters* 86, 335–338 (2003)
- [RV99] Rajagopalan, S., Vazirani, V.: On the bidirected cut relaxation for the metric Steiner tree problem. In: *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms* (1999)
- [RZ05] Robins, G., Zelikovsky, A.: Tighter bounds for graph Steiner tree approximation. *SIAM Journal of Discrete Mathematics* 19, 122–134 (2005)
- [Won84] Wong, R.T.: A dual ascent approach for Steiner trees on a directed graph. *Mathematical Programming* 28, 271–287 (1984)

# Min Sum Edge Coloring in Multigraphs Via Configuration LP

Magnús M. Halldórsson<sup>1</sup>, Guy Kortsarz<sup>2</sup>, and Maxim Sviridenko<sup>3</sup>

<sup>1</sup> School of Computer Science, Reykjavik University, 103 Reykjavik, Iceland  
mmh@ru.is

<sup>2</sup> Department of Computer Science, Rutgers University, Camden, NJ 08102,  
Currently visiting IBM T. J. Watson Research Center  
guyk@camden.rutgers.edu

<sup>3</sup> IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598  
sviri@us.ibm.com

**Abstract.** We consider the scheduling of biprocessor jobs under sum objective (BPSMS). Given a collection of unit-length jobs where each job requires the use of two processors, find a schedule such that no two jobs involving the same processor run concurrently. The objective is to minimize the sum of the completion times of the jobs. Equivalently, we would like to find a sum edge coloring of the given multigraphs, i.e. a partition of its edge set into matchings  $M_1, \dots, M_t$  minimizing  $\sum_{i=1}^t i|M_i|$ .

This problem is APX-hard even in the case of bipartite graphs [M04]. This special case is closely related to the classic open shop scheduling problem. We give a 1.829-approximation algorithm for BPSMS that combines a configuration LP with greedy methods improving the previously best known ratio of 2 [BBH<sup>+</sup>98]. The algorithm uses the fractions derived from the configuration LP and a non-standard randomized rounding. We also give a purely combinatorial and practical algorithm for the case of simple graphs, with a 1.8861-approximation ratio.

**Keywords:** Edge Scheduling, Configuration LP, Approximation Algorithms.

## 1 Introduction

We consider the *biprocessor scheduling* of unit jobs under sum of completion times measure (BPSMS) problem. Given a collection of unit-length jobs where each job requires the use of two processors, find a schedule such that no jobs using the same processors run concurrently. The objective is to minimize the sum of the completion times of the jobs.

The problem can be formalized as an edge coloring problem on multigraphs, where the nodes correspond to processors and edges correspond to the jobs. In each round we schedule a matching from the graph, and the objective is to schedule the edges early on average. This is also known as *sum edge coloring*. Formally, the BPSMS problem is: given a multigraphs  $G$ , find an edge coloring, or a partition into a matchings  $M_1, M_2, \dots$  minimizing  $\sum_i i \cdot |M_i|$ .

Biprocessor scheduling problems have been studied extensively [AB<sup>+</sup>00, KK85, GKMP02, CG<sup>+</sup>85, Y05, GSS87, DK89, K96, FJP01]. Natural application of biprocessor scheduling includes for example file transfer problem in which we have a sender and a receiver (see for example [GKMP02, CG<sup>+</sup>85, Y05]), running two (or more) programs on the same job in order to assure that the result is reliable (see [GSS87]), mutual testing of processors in biprocessor diagnostic links (see [KK85]) and batch manufacturing where jobs simultaneously require two resources for processing (see [DK89]).

We survey some additional previous work on biprocessor scheduling. Kubale [K96] studied the complexity of scheduling biprocessor tasks. He also investigated special classes of graphs, and showed that if jobs have varying length then even non-preemptive biprocessor scheduling of trees is NP-hard. Marx [M04] showed that BPSMS (unit jobs) is NP-hard even in the case of planar bipartite graphs of maximum degree 3 and APX-hard in general bipartite graphs. When the number of processors is constant, Afrati *et al.* [AB<sup>+</sup>00] gave a polynomial time approximation scheme for minimizing the sum of completion times of biprocessor scheduling. Later, their results were generalized in [FJP01] to handle weighted completion times and release times.

The only general approximation result for BPSMS is a 2-approximation algorithm in [BBH<sup>+</sup>98]. Improved bounds of 1.796 [GHKS04] and  $\sqrt{2}$  [GM06] have been given for the special case of bipartite graphs. The bipartite case lends additional importance to BPSMS as it captures a version of the classic open shop scheduling problem and can be denoted as  $O|p_{ij} = 1| \sum C_{ij}$  using the standard scheduling notation [LLRS].

Our main tool in the design of the 1.829 approximation is the solution of a *configuration LP*. A configuration LP is a linear program with a large (usually exponential) number of variables. Usually, it contains a variable corresponding to each “restricted” feasible solution, which in our case corresponds to a feasible assignment of edges to a particular color. See also configuration LP’s for assignment problems [FGMS06, NBY06, BS06, COR01].

## 1.1 Our Results

**Theorem 1.** *The BPSMS problem admits an LP based approximation algorithm with ratio at most 1.829.*

This result holds even if the graph has parallel edges.

This LP-based approach has high polynomial time complexity. Hence, it is of practical interest to study purely combinatorial algorithms.

**Theorem 2.** *The BPSMS problem on graphs with no parallel edges admits a purely combinatorial algorithm whose approximation ratio is at most 1.8861.*

Some of the proofs of propositions are omitted due to lack of space.

## 2 An Overview of the Main Algorithm

We formulate a configuration LP that contains variables  $x_{Mt}$  that indicate whether a matching  $M$  is selected to be scheduled at time  $t$ . We then perform

a randomized rounding, but with a twist. Each of the  $x_{Mt}$  values is chopped up into many tiny fractions that are rounded independently. This ensures a much stronger concentration of the combined rounding. It is possible that an easier rounding scheme (for example choosing each matching  $(M, t)$  independently with probability  $1 - e^{-\alpha x_{Mt}}$  or  $\min\{1, \alpha x_{Mt}\}$ ) would lead to the same result with a slight complication in the analysis. Also, the probability by which a chopped fraction is rounded is multiplied by a universal carefully chosen constant  $\alpha$ . The main reason for using this “chopping” operation is purely technical: we want the expected number of edges incident to a node  $v$  of degree  $d_v$  that are not covered during the first phase to be roughly  $e^{-\alpha}d_v$  even for vertices of small degree. On the other hand we would like to choose an edge on the first rounding phase by exactly  $\alpha$  matchings in expectation.

As a result of the rounding, many matchings “compete” over the same time slot, namely provisionally request to occupy this slot. A careful (and again, randomized) strategy is applied to spread the matchings so that afterwards each round contains at most one matching. The remaining edges are scheduled one by one by a simple greedy algorithm that iteratively schedules the edges in the earliest possible time slot. Even though the greedy algorithm by itself leads to a ratio of 2 (see [BBH<sup>+</sup>98]), we prove that the interaction between the randomized first and greedy second steps gives a 1.829 ratio.

### 3 An Approximation Algorithm Using Configuration LP

Let  $\mathcal{M}$  denote the collection of all (possibly non-maximal) matchings in the graph, including the empty matching. For an edge  $h$ , let  $\mathcal{M}_h$  denote the set of matchings that contain  $h$ . We form an LP with a variable  $x_{Mt}$  for each matching  $M$  and each time slot  $t$ .

$$\begin{aligned} \text{Minimize} \quad & opt^* = \sum_{t, M} t|M|x_{Mt} \\ \text{subject to:} \quad & \sum_{M \in \mathcal{M}} x_{Mt} = 1 && \text{for each } t && (1) \\ & \sum_{M \in \mathcal{M}_h} \sum_t x_{Mt} = 1 && \text{for each edge } h && (2) \\ & x_{Mt} \geq 0 && \text{for each } t \text{ and } M \end{aligned}$$

Equality (1) ensures that in an integral solution each round contains exactly one matching (if a round is not used by the optimum then it is assigned an empty matching). Equality (2) indicates that each edge belongs to exactly one scheduled matching. Since every valid solution must obey these constraints, the linear programming relaxation is indeed a relaxation of our problem.

**Proposition 1.** *The LP can be solved exactly in polynomial time and we may assume the solution vector is a basic feasible solution.*

### 3.1 The Randomized Rounding Algorithm

Observe that the number of rounds in any reasonable schedule is at most  $\Psi = 2\Delta - 1$ , since each edge has at most  $2\Delta - 2$  adjacent edges. Let  $\{x_{Mt}\}$  be the optimum fractional solution for the instance at hand. We assume that  $\{x_{Mt}\}$  is a basic feasible solution.

Let  $1/2 \leq \alpha \leq 1$  be a universal constant whose value will be optimized later. For technical reasons (to be clarified later) we need to do a *chopping operation* described as follows. Each  $x_{Mt}$  is replaced by a set of  $n^4$  smaller fractions  $y_{Mt}^1, \dots, y_{Mt}^{n^4}$ , each with a value of  $x_{Mt}/n^4$ . We think of  $y_{Mt}^j$  as associated with a matching  $M_{j,t} = M$  and “responsible” for an  $x_{Mt}/n^4$  fraction of  $x_{Mt}$ .

The algorithm has a randomized phase followed by a deterministic phase. The random phase first assigns to each time slot  $t$  a *bucket*  $B_t$  of matchings, all provisionally requesting to occupy that slot. Since bucket  $B_t$  may contain more than one matching, a procedure is needed to decide which unique matching to use in each round  $t'$ . In this *spreading* procedure, the matchings are evenly distributed so that each *actual round*  $t$  receives at most one matching, thus forming a proper schedule. Because of the spreading, the actual unique matching assigned to round  $t$  may not even belong to  $B_t$ .

**Remark:** For simplicity of the presentation we present the algorithm on simple graphs (no parallel edges). If there are parallel edges it is possible to get the same ratio by a larger chopping. Details are omitted from this preliminary report.

Formally, the algorithm is defined as follows:

- (i) **The initial ordering step:** We choose an arbitrary *initial order* on all the chopped  $M_{j,t}$  matchings. The order is deterministic and obeys time slots in that if  $t < t'$  then any  $M_{j,t}$  must come in the initial order *before*  $M'_{j',t'}$  regardless of  $j', j, M', M$ .
- (ii) **The randomized rounding step:** For each  $M'_{j,t}$ ,  $1 \leq t \leq \Psi$  and  $1 \leq j \leq n^4$ , place  $M'$  independently at random in bucket  $B_{t'}$  with probability  $\alpha \cdot y_{t',M'}^j$ . Several matchings may be selected into the same bucket  $B_{t'}$ , indicating their provisional wish to be scheduled at time  $t'$ .
- (iii) **The random reordering step:** Randomly reorder all matchings that reached bucket  $B_t$ .  
**Remark:** The random ordering step is completely independent of the randomized rounding step.
- (iv) **The spreading step:** Spread the resulting matchings, in order of bucket number and the chosen order within each bucket. Thus, all matchings of bucket  $B_i$  will be scheduled before all matchings of bucket  $B_{i+1}$ , etc. For each  $i$ , let  $m_i$  denote the number of matchings in  $B_i$ . By increasing bucket numbers  $i$ , the matchings in  $B_i$  are assigned to rounds  $\sum_{j=1}^{i-1} m_j + 1$  through  $\sum_{j=1}^{i-1} m_j + m_i$ , with the ordering of the  $m_i$  matchings that belong to  $B_i$  following the random order of the previous step.
- (v) **The assignment step:** We assign each edge  $h$  to the first matching  $M$  that contains  $h$  (in the order of spreading). If  $M$  is the  $i$ -th matching in the order



then the *finish time* of  $h$  is  $f(h) = i$ . We then remove copies of  $h$  from all other matchings.

Denote by  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_\gamma$  the matchings chosen in this first phase. Let  $E_c = \bigcup_{i=1}^\gamma \mathcal{M}_i$  be the set of edges covered in this first phase, and  $E_u = E - E_c$  be the set of *uncovered* edges.

In the second phase, the edges of  $E_u$  are now scheduled in an arbitrary sequence by a straightforward greedy procedure. Namely, add edges one by one to the current schedule (collection of matchings) as early as possible, i.e. if an edge  $(z, v) \in E_u$  is scheduled at time  $t$  then for every time step  $t' < t$  either edge incident to node  $z$  or an edge incident to node  $v$  is scheduled at time  $t'$ . Let the final schedule be  $\{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_\Phi\}$  where  $\Phi \geq \gamma$ .

### 3.2 Analysis

Recall that for each matching  $M$  and each bucket  $t$ , there are  $n^4$  tries of adding  $M$  into  $t$ . Let  $A_{Mt}^j$  be the event the  $j$ -th try succeeds and  $Y_{Mt}^j$  the indicator random variable of  $A_{Mt}^j$ . Let  $X_{Mt} = \sum_j Y_{Mt}^j$ . Since  $\Pr[A_{Mt}^j] = \alpha y_{Mt}^j$ , by linearity of expectation,  $\mathbf{E}[X_{Mt}] = \alpha \cdot x_{Mt}$ .

Let  $\mathcal{A}_h = \{A_{Mt}^j | j = 1, \dots, n^4, M \in \mathcal{M}_h, t = 1, \dots, \Psi\}$  be the set of events involving edge  $h$ . We order these events for  $h$  according to the initial order (see the first step of the algorithm), and denote  $\mathcal{A}_h = \{A_1^h, A_2^h, \dots, A_{k_h}^h\}$  according to this order. The  $i$ -th event  $A_i^h$  in this order corresponds to some  $A_{M',t'}^j$  that is the  $j$ th trial of some pair  $t', M'$ . Correspondingly, let  $y_i^h$  denote the chopped value  $y_{M',t'}^j$ , namely,  $y_i^h = \Pr[A_{M',t'}^j] / \alpha$ . Note that the order above depends for now only on the initial order and does not depend on the later random ordering in buckets. Let  $k_h$  be the total number of events related to  $h$ .

Let  $\text{deg}(v)$  be the degree of vertex  $v$ , and  $d_c(v)$  be the number of edges incident on  $v$  in  $E_c$  (as a random variable).

Let the *finish time* of an edge  $h$ , denoted  $f(h)$ , be the round  $i$  in which  $h$  was scheduled, i.e., if the unique matching containing  $h$  was  $\mathcal{M}_i$ . We split this quantity into two parts: its *covering contribution*  $f_c(h)$  and its *uncovered contribution*  $f_u(h)$ . If  $h \in E_c$ , then  $f_c(h) = f(h)$  and  $f_u(h) = 0$ , while if  $h \in E_u$ , then  $f_c(h) = 0$  and  $f_u(h) = f(h)$ . Clearly,  $f_c(h) + f_u(h) = f(h)$ . Thus,  $f_c(h)$  ( $f_u(h)$ ) correspond to the amount contributed by  $h$  to the objective function from the first (second) phase, respectively.

**Remark:** Throughout, we assume, without mentioning it explicitly, that  $n$  is large enough (thus larger than any universal constant required in the analysis).

*Proof outline:* We analyze the performance of the algorithm by bounding individually the expected covering and uncovered contribution of each edge. For the former, we use the initial ordering defined on the events involving the edge  $h$  and first evaluate the expected covering contribution conditioned on the edge being selected due to the first event and more generally  $i$ -th event in the ordering, respectively (Proposition 2). The unconditional covering contribution is given in Lemma 1. We first argue tight bounds on the probability of an edge being

uncovered (Proposition 5) and on the expected number of covered edges incident on a vertex (Proposition 8). We then argue the uncovered contribution of each edge (Lemma 2). Finally, summing over all the edges of the graph, we combine the two contributions to derive the approximation in Theorem 4, noting that the part of the ratio corresponding to the covering contributions is in terms of the LP objective, while the uncovered contributions will be in terms of another previously studied lower bound on the optimal value (Definition 2) that is in fact weaker than LP lower bound.

*Proof details:* We first bound  $f_c(h)$ , conditioned on  $h \in E_c$ . Recall that  $\mathcal{A}_h = \{A_1^h, A_2^h, \dots, A_{k_h}^h\}$  is the initial ordering of matchings containing  $h$ . At least one of the events in this list has succeeded.

We compute the finish time of  $h$  under the assumption that events  $A_1^h, \dots, A_p^h$  failed and the first to succeed was  $A_{p+1}^h$ , for some  $p \geq 0$ .

**Proposition 2.** *The expected covering contribution of edge  $h$ , given that the first event in  $\mathcal{A}_h$  that occurred is  $A_{p+1}^h$ , is bounded by*

$$\mathbf{E} \left[ f_c(h) \mid \bigcap_{i \leq p} \overline{A_i^h} \cap A_{p+1}^h \right] \leq \alpha \cdot \left( t_{p+1}^h - \frac{1}{2} \sum_{i=1}^{p+1} y_i^h + \frac{1}{\alpha} - \frac{1}{2} \right).$$

*Proof.* We bound the expected waiting time of an edge, or the expected number of matchings that precede its round. We first bound the number of those coming from earlier buckets, and then those coming from the same bucket as  $h$ .

The expected sum of fractions chosen per bucket is  $\alpha \cdot \sum_{M \in \mathcal{M}} x_M t = \alpha$ . Thus the unconditional expected waiting time of  $h$  due to the  $t_{p+1}^h - 1$  first rounds is  $\alpha(t_{p+1}^h - 1)$ . However, we are given that previous events involving  $h$  did not occur. Let  $B = \{A_i^h, i \leq p \mid t_i^h < t_{p+1}^h\}$  be the set of events concerning  $h$  that belong to earlier buckets. Then,  $h$  waits

$$\alpha \left( t_{p+1}^h - 1 - \sum_{A_i^h \in B} y_i^h \right) \tag{3}$$

in expectation for previous buckets.

We now consider the matchings belonging to the current bucket  $t_{p+1}^h$ . Let  $W = \{A_i^h \mid i \leq p, t_i^h = t_{p+1}^h\}$  be the set of events involving  $h$  that precede it in the initial order but also concern the same bucket. The expected number of matchings in bucket  $t_{p+1}^h$ , conditioned on  $\bigcap_{A \in W} \overline{A}$  (i.e. none of its preceding events involving  $h$  occurring), equals

$$\alpha \left( 1 - \sum_{A_i^h \in W} y_i^h \right).$$

This amount is independent of the random ordering step, but the matchings will be spread within the bucket randomly. Hence, taking the expectation also over the random orderings, the waiting time of  $h$  for this bucket is at most

$$\alpha \left( 1/2 - \sum_{A_i^h \in W} y_i^h/2 - y_{p+1}^h/2 \right). \tag{4}$$

We now add the waiting times of (3) and (4), observing that worst case occurs when  $B = \emptyset$ . In that case the expected waiting time is bounded by

$$\alpha \left( t_{p+1}^h - \frac{1}{2} - \sum_{i=1}^{p+1} y_i^h/2 \right).$$

Adding the round of  $M_{p+1}^h$  itself yields the claim.

**Remark:** The above number can indeed be strictly larger than the round of  $h$ . The round of  $h$  can be smaller if the following events happen. There is a matching containing  $h$  located *after*  $M_{p+1}^h$  in the initial ordering, this matching reaches slot  $t_{p+1}^h$  and is located before  $M_{p+1}^h$  by the random ordering. However, it seems hard to use this fact to improve the ratio.

**Proposition 3.** *For any non-negative numbers  $y_1, y_2, \dots, y_k$  satisfying  $\sum_{i=1}^k y_i = 1$ , it holds that*

$$\frac{y_1^2}{2} + y_2 \cdot \left( y_1 + \frac{y_2}{2} \right) + y_3 \left( y_1 + y_2 + \frac{y_3}{2} \right) + \dots + y_k \cdot \sum_{i=1}^{k-1} y_i + \frac{y_k^2}{2} = \frac{1}{2}. \tag{5}$$

*Proof.* Denote the left hand side by  $S$  and rearrange its terms to obtain

$$S = y_1 \left( \frac{y_1}{2} + y_2 + \dots + y_k \right) + y_2 \left( \frac{y_2}{2} + y_3 + \dots + y_k \right) + \dots + y_k \left( \frac{y_k}{2} \right).$$

Applying the equality  $\sum_{i=1}^k y_i = 1$  to each of the parenthesized sums, we have that

$$\begin{aligned} S &= y_1 \left( 1 - \frac{y_1}{2} \right) + y_2 \left( 1 - y_1 - \frac{y_2}{2} \right) + \dots + y_k \left( 1 - y_1 - y_2 - \dots - y_{k-1} - \frac{y_k}{2} \right) \\ &= \sum_{i=1}^k y_i - S = 1 - S. \end{aligned}$$

Hence,  $S$  is  $1/2$ , as claimed.

The next claim appeared many times in the approximation algorithms literature. Chudak and Shmoys [CS03] were the first ones to use it, see also [FGMS06] for a simpler proof.

**Proposition 4.** *Let  $t_1, \dots, t_k$  be positive numbers so that  $t_1 \leq t_2 \leq \dots \leq t_k$ , and  $x_1, \dots, x_k$  be positive numbers such that  $\sum_{i=1}^k x_i \leq 1$ . Then:*

$$t_1 \cdot x_1 + t_2 \cdot (1 - x_1) \cdot x_2 + \dots + \prod_{i=1}^{k-1} (1 - x_i) \cdot x_k t_k \leq \frac{\left( 1 - \prod_{i=1}^k (1 - x_i) \right) \cdot \sum_{i=1}^k t_i x_i}{\sum_{i=1}^k x_i}.$$

**Proposition 5.** *The probability that an edge  $h$  is not covered is bounded by*

$$\frac{1}{e^\alpha} \left(1 - \frac{3}{n^2}\right) \leq Pr[h \in E_u] = \prod_{i=1}^{k_h} (1 - \alpha y_i^h) \leq \frac{1}{e^\alpha}.$$

**Remark:** The r.h.s inequality follows easily by convexity. But the l.h.s is complex to prove. The proof of the l.h.s. inequality strongly uses the chopping, and also uses that  $x_{Mt}$  is a basic feasible solution hence contains at most  $|E| + 2\Delta - 1 < 2n^2$  non-zero entries. For lack of space this proof is left to the full version.

**Notation 3.** Let  $b_i^h = \alpha \cdot \left( t_j^h - \left( \sum_{j=1}^i y_j^h \right) / 2 + 1/\alpha - 1/2 \right)$ .

**Lemma 1.** *The expected covering contribution of an edge  $h$  is bounded by*

$$\mathbf{E}[f_c(h)] \leq \left(1 + \frac{1}{n}\right) \cdot \left(1 - \frac{1}{e^\alpha}\right) \cdot \left( \left(1 - \frac{3\alpha}{4}\right) + \alpha \sum_{t, M \in \mathcal{M}_h} x_{Mt} \cdot t \right).$$

*Proof.* By Proposition 2  $\mathbf{E} \left[ f_c(h) \mid \bigcup_{j=1}^{i-1} \overline{A}_i^h \cup A_i^h \right] \leq b_i^h$ . Thus,

$$\begin{aligned} \mathbf{E}[f_c(h)] &= \mathbf{E} \left[ f_c(h) \mid A_1^h \right] \cdot Pr(A_1^h) + \mathbf{E} \left[ f_c(h) \mid \overline{A}_1^h \cap A_2^h \right] \cdot Pr(\overline{A}_1^h \cap A_2^h) + \dots \\ &\quad + \mathbf{E} \left[ f_c(h) \mid \bigcap_{i < k_h} \overline{A}_i^h \cap A_{k_h}^h \right] \cdot Pr(\bigcap_{i < k_h} \overline{A}_i^h \cap A_{k_h}^h) + 0 \cdot Pr(\bigcap_{i \leq k_h} \overline{A}_i^h) \\ &\leq \alpha \cdot y_1^h \cdot b_1^h + \left(1 - \alpha \cdot y_1^h\right) \cdot \alpha \cdot y_2^h \cdot b_2^h + \left(1 - \alpha \cdot y_1^h\right) \cdot \left(1 - \alpha \cdot y_2^h\right) \cdot \alpha y_3^h \cdot b_3^h \\ &\quad + \dots + \prod_{i=1}^{k-1} \left(1 - \alpha \cdot y_i^h\right) \alpha \cdot y_k^h \cdot b_k^h \\ &\leq \left(1 - \left(\prod_{i=1}^k \left(1 - \alpha \cdot y_i^h\right)\right)\right) \cdot \sum_{i=1}^k y_i^h b_i^h. \quad (\text{By Proposition 4}) \end{aligned}$$

The  $\alpha$  term that multiplied every term before the last inequality was canceled because  $\sum_{i=1}^k \alpha y_i^h = \alpha$ .

The lemma now follows from the combination of the following two propositions.

**Proposition 6.**  $\sum_{i=1}^k y_i^h b_i^h \leq \alpha \sum_{t, M \in \mathcal{M}} x_{Mt} \cdot t + (1 - 3\alpha/4)$ .

*Proof.* Recall that

$$y_i^h b_i^h = \alpha y_i^h t_i^h + \alpha \cdot \left( y_i^h \cdot \left( -\frac{1}{2} \sum_{j=1}^i y_j^h + \frac{1}{\alpha} - \frac{1}{2} \right) \right).$$

We break the sum into three parts, and first analyze the total contribution of the two last terms,  $-\alpha \cdot y_i^h \left( \sum_{j=1}^i y_j^h \right) / 2$ , and  $\alpha \cdot y_i^h (1/\alpha - 1/2)$  when summing over all  $i$ . By Proposition [3](#)

$$-\alpha \sum_{i=1}^{k_h} y_i^h \cdot \left( \frac{1}{2} \sum_{j=1}^i y_j^h \right) \leq -\frac{\alpha}{2} \sum_{i=1}^{k_h} y_i^h \cdot \left( \sum_{j=1}^{i-1} y_j^h + \frac{y_i^h}{2} \right) = -\alpha/4, \tag{6}$$

Since  $\sum_{i=1}^{k_h} y_i^h = 1$  we have that

$$\alpha \left( \frac{1}{\alpha} - \frac{1}{2} \right) \cdot \sum_{i=1}^{k_h} y_i^h = \left( 1 - \frac{\alpha}{2} \right). \tag{7}$$

Finally, consider the sum of the terms  $\alpha y_i^h t_i^h$ . By re-indexing, we have that

$$\sum_{i=1}^{k_h} \alpha y_i^h \cdot t_i^h = \alpha \sum_{t, M \in \mathcal{M}_h} \sum_{i=1}^{n^4} y_{Mt}^i \cdot t = \alpha \sum_{t, M \in \mathcal{M}_h} x_{Mt} \cdot t. \tag{8}$$

Adding [\(6\)](#), [\(7\)](#) and [\(8\)](#) now yields the claim.

**Proposition 7.**  $\left( 1 - \left( \prod_{i=1}^{k_h} (1 - \alpha \cdot y_i^h) \right) \right) \leq \left( 1 + \frac{1}{n} \right) \cdot \left( 1 - \frac{1}{e^\alpha} \right).$

*Proof.* The term is maximized when  $\prod_{i=1}^{k_h} (1 - \alpha \cdot y_i^h)$  is minimized. Using

$$\frac{1}{e^\alpha} \left( 1 - \frac{3}{n^2} \right) \leq \prod_{i=1}^{k_h} (1 - \alpha y_i^h)$$

(Proposition [5](#)), we get

$$\left( 1 - \left( \prod_{i=1}^{k_h} (1 - \alpha \cdot y_i^h) \right) \right) \leq \left( 1 - \frac{1}{e^\alpha} \cdot \left( 1 - \frac{3}{n^2} \right) \right) \leq \left( 1 + \frac{1}{n} \right) \cdot \left( 1 - \frac{1}{e^\alpha} \right)$$

The last inequality uses our assumption that  $\alpha \geq 1/2$ .

We now deal with the possibility that  $h \in E_u$  and bound its expected contribution.

**Proposition 8.** *The expected number of covered edges incident on a vertex  $v$  is bounded by*

$$\mathbf{E}[d_c(v)] \leq \left( 1 + \frac{1}{n} \right) \left( 1 - \frac{1}{e^\alpha} \right) \text{deg}(v).$$

*Proof.* By Proposition 5, we have for every  $h \in E(v)$  that

$$\Pr[h \in E_c] = 1 - \Pr[h \in E_u] \leq 1 - \frac{1}{e^\alpha} \left(1 - \frac{3}{n^2}\right) = \left(1 - \frac{1}{e^\alpha}\right) + \frac{3}{n^2 \cdot e^\alpha}.$$

Using  $\alpha \geq 1/2$ , we have that, for large enough  $n$ ,

$$\Pr[h \in E_c] \leq \left(1 + \frac{1}{n}\right) \left(1 - \frac{1}{e^\alpha}\right).$$

By linearity of expectation,

$$\mathbf{E}[d_c(v)] \leq \left(1 + \frac{1}{n}\right) \left(1 - \frac{1}{e^\alpha}\right) \text{deg}(v).$$

**Remark:** Without the chopping operation it seems difficult to bound the expectation of  $d_c(v)$  as above, especially considering that some of the degrees may be small. In fact if there are many parallel edges that chopping has to be into smaller pieces.

We are ready to bound from above the expectation of the sum of finish times in the greedy phase. Consider an uncovered edge  $(u, v) \in E_u$ . This edge would have to first wait a total of at most  $d_c(v) + d_c(u)$  rounds until all covered edges incident on  $v$  and  $u$  are scheduled. After that, each time the edge  $(u, v)$  is not selected, one of  $u$  or  $v$  or both are matched. Thus, each time the edge waits can be charged to either an edge of  $u$  or an edge of  $v$  that belongs to  $E_u$ . Thus, the contribution of the greedy step to the sum of finish times is at most:

$$\sum_{v \in V} \sum_{i=d_c(v)+1}^{\text{deg}(v)} i = \left( \sum_{i=1}^{\text{deg}(v)} i - \sum_{i=1}^{d_c(v)} i \right) = \sum_{v \in V} \left( \binom{\text{deg}(v)+1}{2} - \binom{d_c(v)+1}{2} \right). \tag{9}$$

Now, divide equally the term  $\binom{\text{deg}(v)+1}{2} - \binom{d_c(v)+1}{2}$ , charging an equal amount to each edge in  $E_u(v)$ . Then, the edge  $h = (z, v)$  is charged

$$\begin{aligned} & \frac{\text{deg}(z)(\text{deg}(z)+1) - d_c(z)(d_c(z)+1)}{2(\text{deg}(z) - d_c(z))} + \frac{\text{deg}(v)(\text{deg}(v)+1) - d_c(v)(d_c(v)+1)}{2(\text{deg}(v) - d_c(v))} \\ &= \frac{\text{deg}(z) + d_c(z)}{2} + \frac{\text{deg}(v) + d_c(v)}{2} + 1. \end{aligned}$$

**Definition 1.** Define

$$f'_u(h) = \frac{\text{deg}(z) + d_c(z)}{2} + \frac{\text{deg}(v) + d_c(v)}{2} + 1$$

if  $h \in E_u$  and 0 otherwise.

Observe that  $\sum_{h \in E} f'_u(h) \geq \sum_{h \in E} f_u(h)$ . Hence, it is enough to bound the expectation of  $f'_u(h)$ , for the purpose of evaluating the sum of the expected values.

**Lemma 2.** *The contribution of an uncovered edge  $h$  is given by*

$$\mathbf{E} [f'_u(h)] \leq \left(1 + \frac{1}{n}\right) \cdot \frac{1}{e^\alpha} \left[ \left(2 - \frac{1}{e^\alpha}\right) \cdot \left(\frac{\text{deg}(z) + \text{deg}(v)}{2} + 1\right) - \left(1 - \frac{1}{e^\alpha}\right) \right]$$

*Proof.* The probability that  $h$  is uncovered is at most  $1/e^\alpha$  (see Proposition 5). Thus,

$$\begin{aligned} \mathbf{E} [f'_u(h)] &\leq \frac{1}{e^\alpha} \cdot \mathbf{E} \left[ \frac{\text{deg}(z) + d_c(z)}{2} + \frac{\text{deg}(v) + d_c(v)}{2} + 1 \right] \\ &\leq \frac{1}{e^\alpha} \left(1 + \frac{1}{n}\right) \left(2 - \frac{1}{e^\alpha}\right) \left(\frac{\text{deg}(z) + \text{deg}(v)}{2} + 1\right) \\ &\quad - \frac{1}{e^\alpha} \left(1 + \frac{1}{n}\right) \left(1 - \frac{1}{e^\alpha}\right). \quad (\text{By Proposition 8}) \end{aligned}$$

Each edge  $h'$  contributes two terms that depend *only on  $n$  and  $\alpha$* . There is the positive contribution (see Proposition 11)

$$\left(1 + \frac{1}{n}\right) \cdot \left(1 - \frac{1}{e^\alpha}\right) \cdot \left(1 - \frac{3\alpha}{4}\right) \tag{10}$$

and the negative term from Lemma 2

$$-\frac{1}{e^\alpha} \cdot \left(1 + \frac{1}{n}\right) \cdot \left(1 - \frac{1}{e^\alpha}\right). \tag{11}$$

This amounts to

$$\left(1 + \frac{1}{n}\right) \cdot \left(1 - \frac{1}{e^\alpha}\right) \cdot \left(1 - \frac{3\alpha}{4} - \frac{1}{e^\alpha}\right). \tag{12}$$

For  $\alpha \geq 0.606$ , the above term is negative and only makes the upper bound smaller. We shall henceforth assume that  $\alpha$  satisfies this property and will ignore these two terms in the analysis.

The following measure has been useful for lower bounding the cost of the optimal solution.

**Definition 2.**  $q(G) = \sum_{v \in V} \binom{\text{deg}(v) + 1}{2}$ .

Let  $opt(G)$  denote the cost of the optimal sum edge coloring of  $G$ , and recall that  $opt^*$  denotes the value of the linear program. It was shown in [BBH<sup>+</sup>98] that  $opt(G) \geq q(G)/2$ .

**Proposition 9.**  $opt(G) \geq q(G)/2 = \sum_{(z',v') \in E} \left(\frac{\text{deg}(z') + \text{deg}(v')}{4} + \frac{1}{2}\right)$

**Theorem 4.** *The expected approximation ratio of the algorithm is at most 1.829.*

*Proof.* Ignoring the terms (I0) and (II) we get:

$$\begin{aligned}
& E \left[ \sum_h f(h) \right] \\
&= \sum_h (\mathbf{E}[f_c(h)] + \mathbf{E}[f_u(h)]) = \sum_h (\mathbf{E}[f_c(h)] + \mathbf{E}[f'_u(h)]) \\
&\leq \sum_h \left( \left(1 + \frac{1}{n}\right) \alpha \cdot \left(1 - \frac{1}{e^\alpha}\right) \sum_{t, M \in \mathcal{M}_h} x_{tM} \cdot t \right) \quad (\text{By Lemma 1}) \\
&+ \sum_{h=(z,v)} \left( \left(1 + \frac{1}{n}\right) \frac{2}{e^\alpha} \left(2 - \frac{1}{e^\alpha}\right) \cdot \left(\frac{\deg(z) + \deg(v)}{4} + \frac{1}{2}\right) \right) \quad (\text{By Lemma 2}) \\
&= \left(1 + \frac{1}{n}\right) \alpha \left(1 - \frac{1}{e^\alpha}\right) \text{opt}^* + \left(1 + \frac{1}{n}\right) \frac{2}{e^\alpha} \left(2 - \frac{1}{e^\alpha}\right) \cdot \frac{q(G)}{2} \\
&\leq \left(1 + \frac{1}{n}\right) \cdot \left(\alpha \cdot \left(1 - \frac{1}{e^\alpha}\right) + \frac{2}{e^\alpha} \left(2 - \frac{1}{e^\alpha}\right)\right) \text{opt}(G) \quad (\text{By Proposition 9})
\end{aligned}$$

For  $\alpha = 0.895$ , this gives a ratio of at most 1.828. Thus, the Proposition follows for large enough  $n$ .

## 4 Combinatorial Approach

We use the algorithm ACS of [HKS03]. This algorithm does not seem to be suited for graphs with parallel edges as we do not have a strong enough version of Vizing's theorem for such graphs. This algorithm is designed to operate in rounds, where in each round it finds a collection of matchings. A  $k$ -matched set is a collection of edges partitioned into  $k$  matchings. Let  $\beta_k(G)$  denote the maximum total weight of a  $k$ -matched set in  $G$ .

In each round  $i$ , the algorithm seeks a maximum  $k_i$ -matched set,  $\text{Mat}(G, k_i)$ , where  $k_1, k_2, \dots$  forms a geometric sequence. The base  $q$  of the sequence is given as parameter, while the offset  $\alpha$  is selected uniformly at random from the range  $[0, 1)$ .

### ACS( $G, q$ )

```

 $\alpha = \mathbf{U}[0, 1); i \leftarrow \ell_0 \leftarrow 0$ 
while ( $G \neq \emptyset$ ) do
   $k_i = \lfloor q^{i+\alpha} \rfloor$ 
   $(M_{\ell+1}, M_{\ell+2}, \dots, M_{\ell+k_i}) \leftarrow \text{Mat}(G, k_i)$ 
   $G \leftarrow G - \cup_i M_i$ 
   $i \leftarrow i + 1; \ell_i \leftarrow \ell_{i-1} + k_{i-1}$ 
end
```

This algorithm attains a performance ratio of 1.796 when supplied with an exact algorithm  $\text{Mat}(G, k)$  for finding  $k$ -matched sets [HKS03]. This led to a



1.796-approximation of BPSMS in bipartite graphs [GHKS04]. In general line graphs, it is not possible to find an optimal set of matchings. Instead, we look for a dual approximation, consisting of a  $b_i$ -matched set of weight at least  $\beta_{k_i}$ , where  $b_i$  is not much larger than  $k_i$ . If  $b_i/k_i < c$ , for some constant  $c$ , this leads to a  $1.796 \cdot c$ -approximation.

If the graph has no parallel edges, we can obtain a dual approximation of the  $k$ -matched set problem with a unit additive error. Recall that a  $b$ -matching is a subgraph where each nodes has at most  $b$  incident edges. It can be found in polynomial time by a reduction to matching (cf. [CCPS98]). Our algorithm first finds a maximum  $k_i$ -matching, and then uses Vizing’s algorithm to partition it into  $k_i + 1$  matchings. Only when  $k_i = 1$ , we already have a single matching and need not pay the additional unit term.

First, we give a tighter bound of the additive term in the cost analysis of each vertex. We obtain the following refinement of Lemma 2.4 of [HKS03] (i.e. an improvement in the additive factor). Let  $\psi_1(v)$  denote the color of vertex  $v$  assigned by ACS, when using an optimal oracle for  $k$ -matched set, and assuming that with probability  $1/2$  the matchings for each block are reordered. Clearly, the final output that does not reorder the blocks will be of no larger total cost.

**Lemma 3.** *For  $q \approx 3.591$ , the solution to  $\ln x = (x + 1)/x$ , we have that*

$$\mathbf{E}[\psi_1(v)] \leq 1.796 \psi_{\text{OPT}}(v) - 0.710.$$

The other issue in the analysis is to limit the cost incurred by the additional matchings introduced by the Vizing colorings. The number of new matchings that delay a vertex  $v$  is equal to the number of non-unit sized blocks preceding  $v$ ’s block, plus a half for the increase in  $v$ ’s own block.

**Lemma 4.** *Let  $v$  be a vertex of optimal color  $\psi_{\text{OPT}}(v) = x$ ,  $x > 1$ . Then, the expected extra cost to  $v$  due to the Vizing coloring is at most  $\log_q(x/2) + 1/2$ .*

*Proof.* The number  $h_v$  of the block in which  $v$  is colored is given by

$$h_v = 1 + \lfloor \log_q x \rfloor + t_v,$$

where  $t_v$  is a Bernoulli variable with probability  $\log_q x - \lfloor \log_q x \rfloor$ . Thus,  $\mathbf{E}[h_v] = 1 + \log_q x$ .

The first block may or may not be of unit size, and in the latter case no additional matching is needed. The probability that the first block is of unit size is the probability that  $q^\alpha < 2$ , or the probability that  $\alpha < \log_q 2$ , which is  $\log_q 2$ . Finally, the last block contributes only one a cost of  $1/2$ . Hence, the additional cost for  $v$  is bounded by  $1 + \log_q x - \log_q 2 - 1/2 = \log_q(x/2) + 1/2$ .

We now combine these observations to bound the performance of our algorithm.

**Theorem 5.** *The performance ratio of our algorithm is at most 1.8861. This is on a per vertex basis, and thus applies, e.g., also to the weighted case.*

*Proof.* Keeping in mind that the cost of the algorithm is at most  $\sum_v \mathbf{E}[\psi_1(v)]$ , we shall be bounding the amortized expected cost of  $\psi_1(v)$  to the optimal cost  $x = \psi_{\text{OPT}}(v)$  of each vertex  $v$ . If  $x = 1$ , the extra cost is  $(1 - \log_q 2) \cdot 1/2 = 0.2289$ . Then, the expected cost  $a(1)$  is at most  $1.796x - 0.710 + 0.23 \leq 1.316$ . For  $x > 1$ , the expected cost of coloring  $v$  is given by

$$a(x) = \mathbf{E}[\psi_1(v)] + \log_q(x/2) + 1/2 \leq 1.796x + \log_q(x/2) - 0.210.$$

The function  $a(x)/x$  is maximized when  $x = 7$ , yielding a performance ratio of at most 1.90599.

We can improve this bound as follows. Divide the optimal coloring into a set of *slabs*. Let  $t_{\text{opt}}$  be the number of matchings in the optimal solutions, and let  $w(M_i)$  be the weight of the  $i$ -th matching,  $i = 1, \dots, t_{\text{opt}}$ . Define  $s_{t_{\text{opt}}+1} = 0$  and  $s_i = w(M_i) - w(M_{i+1})$ , for  $i = 1, \dots, t_{\text{opt}}$ . We have that

$$\text{OPT} = \sum_{i=1}^{t_{\text{opt}}} \sum_{j=1}^i j \cdot s_i.$$

Thus, we can view the instance as a union of  $t_{\text{opt}}$  subgraphs, or slabs, where the  $i$ -th subgraph consists of  $i$  matchings each of weight  $s_i$ . We bound the performance ratio of each slab separately, and use the largest to obtain a bound on the overall performance.

The cost of our algorithm on slab  $i$  equals  $s_i \sum_{j=1}^i a(j)$ , while the optimal cost is  $\sum_{j=1}^i j \cdot s_i$ . Evaluating these bounds computationally shows that their ratio increases until  $i = 15$  after which it decreases. The maximum ratio is then bounded by 1.8861.

## References

- [AB<sup>+</sup>00] Afrati, F., Bampis, E., Fishkin, A., Jansen, K., Kenyon, C.: Scheduling to minimize the average completion time of dedicated tasks. In: Kapoor, S., Prasad, S. (eds.) FST TCS 2000. LNCS, vol. 1974, pp. 454–464. Springer, Heidelberg (2000)
- [BBH<sup>+</sup>98] Bar-Noy, A., Bellare, M., Halldórsson, M.M., Shachnai, H., Tamir, T.: On chromatic sums and distributed resource allocation. *Inf. Comp.* 140, 183–202 (1998)
- [BS06] Bansal, N., Sviridenko, M.: The Santa Claus problem. In: STOC, pp. 31–40 (2006)
- [CCPS98] Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A.: *Combinatorial Optimization*. John Wiley and Sons, Chichester (1998)
- [CG<sup>+</sup>85] Coffman Jr., E.G., Garey, M.R., Johnson, D.S., LaPaugh, A.S.: Scheduling file transfers. *SIAM J. Comput.* 14, 744–780 (1985)
- [COR01] Chuzhoy, J., Ostrovsky, R., Rabani, Y.: Approximation Algorithms for the Job Interval Selection Problem and Related Scheduling Problems. In: FOCS, pp. 348–356 (2001)
- [CS03] Chudak, F., Shmoys, D.: Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.* 33(1), 1–25 (2003)

- [DK89] Dobson, G., Karmarkar, U.: Simultaneous resource scheduling to minimize weighted flow times. *Oper. Res.* 37, 592–600 (1989)
- [FJP01] Fishkin, A.V., Jansen, K., Porkolab, L.: On minimizing average weighted completion time of multiprocessor tasks with release dates. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) *ICALP 2001*. LNCS, vol. 2076, pp. 875–886. Springer, Heidelberg (2001)
- [FGMS06] Fleischer, L., Goemans, M., Mirrokni, V., Sviridenko, M.: Tight approximation algorithms for maximum general assignment problems. In: *SODA*, pp. 611–620 (2006)
- [GHKS04] Gandhi, R., Halldórsson, M.M., Kortsarz, G., Shachnai, H.: Approximating non-preemptive open-shop scheduling and related problems. In: *ICALP*, pp. 658–669 (2004)
- [GM06] Gandhi, R., Mestre, J.: Combinatorial algorithms for data migration. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) *APPROX 2006 and RANDOM 2006*. LNCS, vol. 4110. Springer, Heidelberg (2006)
- [GKMP02] Giaro, K., Kubale, M., Malafejski, M., Piwakowski, K.: Dedicated scheduling of biprocessor tasks to minimize mean flow time. In: Wyrzykowski, R., Dongarra, J., Paprzycki, M., Waśniewski, J. (eds.) *PPAM 2001*. LNCS, vol. 2328, pp. 87–96. Springer, Heidelberg (2002)
- [GSS87] Gehring, E.F., Siewiorek, D.P., Segall, Z.: *Parallel Processing: The Cm\* Experience*. Digital Press, Bedford (1987)
- [HKS03] Halldórsson, M.M., Kortsarz, G., Shachnai, H.: Sum coloring interval and  $k$ -claw free graphs with application to scheduling dependent jobs. *Algorithmica* 37, 187–209 (2003)
- [J2001] Jain, K.: A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica* 21, 39–60 (2001)
- [K96] Kubale, M.: Preemptive versus non-preemptive scheduling of biprocessor tasks on dedicated processors. *European J. Operational Research* 94, 242–251 (1996)
- [Kh80] Khachiyan, L.: Polynomial-time algorithm for linear programming. *USSR Comput. Math. and Math. Physics* 20(1), 53–72 (1980)
- [KK85] Krawczyk, H., Kubale, M.: An approximation algorithm for diagnostic test scheduling in multicomputer systems. *IEEE Trans. Comput.* 34, 869–872 (1985)
- [LLRS] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and scheduling: Algorithms and complexity. In: *Handbook in Operations Research and Management Science*, vol. 4, pp. 445–522. North-Holland, Amsterdam (1993)
- [M04] Marx, D.: Complexity results for minimum sum edge coloring (manuscript) (2004)
- [NBY06] Nutov, Z., Beniaminy, I., Yuster, R.: A  $(1-1/e)$ -approximation algorithm for the generalized assignment problem. *Oper. Res. Lett.* 34(3), 283–288 (2006)
- [V64] Vizing, V.G.: On an estimate of the chromatic class of a  $p$ -graph. *Diskrete Analiz* 3, 23–30 (1964)
- [Y05] Kim, Y.-A.: Data migration to minimize the total completion time. *J. of Algorithms* 55, 42–57 (2005)

# An Improved Algorithm for Finding Cycles Through Elements

Ken-ichi Kawarabayashi\*

National Institute of Informatics,  
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan  
k\_keniti@nii.ac.jp

**Abstract.** We consider the following problem: Given  $k$  independent edges in  $G$ . Is there a polynomial time algorithm to decide whether or not  $G$  has a cycle through all of these edges? If the answer is yes, detect such a cycle in polynomial time.

This problem can be viewed as an algorithmic aspect of the conjecture of Lovász [22] and Woodall [34]. For fixed  $k$ , it follows from the seminal result of Robertson and Seymour [29] that there is a polynomial time algorithm to decide this problem. But, the proof of its correctness requires the full power of machinery from the graph minor series of papers, which consist of more than 20 papers and  $> 500$  pages. In addition, the hidden constant is an extremely rapidly growing function of  $k$ . Even  $k = 3$ , the algorithm is not practical at all.

Our main result is to give a better algorithm for the problem in the following sense.

1. Even when  $k$  is a non-trivially super-constant number (up to  $O((\log \log n)^{1/10})$ ), there is a polynomial time algorithm for the above problem (So the hidden constant is not too large).
2. The time complexity is  $O(n^2)$ , which improves Robertson and Seymour's algorithm whose time complexity is  $O(n^3)$ .

Our algorithm has several appealing features. Although our approach makes use of several ideas underlying the Robertson and Seymour's algorithm, our new algorithmic components allow us to give a self-contained proof within 10 pages, which is much shorter and simpler than Robertson and Seymour's. In addition, if an input is a planar graph or a bounded genus graph, we can get a better bound for the hidden constant. More precisely, for the planar case, when  $k$  is a non-trivially super-constant number up to  $k \leq O((\log n / (\log \log n))^{1/4})$ , there is a polynomial time algorithm, and for the bounded genus case, when  $k$  is a non-trivially super-constant number up to  $k \leq O((\log(n/g) / (\log \log(n/g)))^{1/4})$ , there is a polynomial time algorithm, where  $g$  is the Euler genus.

## 1 Introduction

Dirac [7] proved the well-known theorem, which says that, given  $k$  vertices in a  $k$ -connected graph  $G$ ,  $G$  has a cycle through all of them. Starting with this

---

\* Research partly supported by JSPS Postdoctoral Fellowship for Research Abroad.

result, many papers have appeared in this context, c.f, Bondy and Lovász [4], Holton et al. [11], and Kawarabayashi [18], etc. Since 1960's, cycles through a vertex set or an edge set are one of central topics in all of graph theory.

If  $L$  is a set of  $k$  independent edges in a  $k$ -connected graph  $G$  with  $k$  being odd, such that  $L$  is a cut in  $G$ , then clearly  $G$  has no cycles through all the edges of  $L$ . Lovász [22] and Woodall [34], independently, conjectured that this is the only obstruction for  $k$ -connected graphs to have a cycle through  $k$  independent edges.

This conjecture, which is called Lovász-Woodall conjecture, had been known to be true for  $k \leq 5$ . For  $k \leq 2$ , it is easy. Lovász [23] proved the case of  $k = 3$ . Erdős and Györi [8] and Lomonosov [21], independently, proved the case of  $k = 4$ . Sanders [32] proved the case of  $k = 5$ .

Partial general results were due to Woodall [34] and Thomassen [33]. Häggkvist and Thomassen [10] proved the following general result toward the conjecture:

If  $L$  is a set of  $k$  independent edges in a  $(k + 1)$ -connected graph  $G$ , then there is a cycle through all the edges in  $L$ .

This implies a conjecture by Berge [2].

Recently, the current author has already settled the conjecture, and the paper [14] is the first step. The proof is lengthy and complicated. There will be another three papers for the proof, see the references [15,16,17].

One natural question arising from Lovász-Woodall conjecture in terms of an algorithmic view is the following:

Is there a polynomial time algorithm to decide whether or not  $G$  has such a cycle? If the answer is yes, find one in polynomial time.

Actually, if  $k$  is fixed, it follows from the seminal result of Robertson and Seymour on the disjoint paths problem [29] that there is a polynomial time algorithm to decide this problem. Let us remind that the disjoint paths problem is that we are given a graph  $G$ , and a set of  $k$  pairs of vertices in  $G$  (the terminals), then we have to decide whether or not there are mutually disjoint paths connecting given pairs of terminals.

However, there are two drawbacks concerning this algorithm:

First, the proof of the correctness of Robertson-Seymour's algorithm requires a decomposition theorem capturing the structure of all graphs excluding a fixed minor. This is actually the heart of the Graph Minor Theory. It needs the full power of machinery from the graph minor series of papers, and thus runs to several hundred pages [29].

Secondly, the hidden constant is an extremely rapidly growing function of  $k$ . Even  $k = 3$ , the algorithm is not practical at all. It is believed to have very large bounds. (To quote David Johnson [13], "for any instance  $G = (V, E)$  that one could fit into the known universe, one would easily prefer  $|V|^{70}$  to even constant time, if that constant had to be one of Robertson and Seymour's." He estimates one constant in an algorithm for testing for a fixed minor  $H$  to be roughly  $2 \uparrow 2^{2^{2^{\uparrow(2 \uparrow \Theta(|V(H)|))}}}}$ , where  $2 \uparrow n$  denotes a tower  $2^{2^{2^{\dots}}}$  involving  $n$  2's).

So one natural question is: Is there a better algorithm for the problem in terms of the above two issues? Toward this question, LaPaugh and Rivest [20] gave a linear time algorithm for the question when  $k = 3$ , which involves no large hidden constants. Gabow [9] has used this result to find a long cycle in polynomial time. But as far as we are aware, no simple polynomial time algorithms for the above question has been known until now.

Our main theorem is to give an answer for this question. Specifically, here is our main result.

**Theorem 1.** *There is a polynomial time algorithm to decide whether or not there is a cycle through  $k$  independent edges on an arbitrary  $n$ -vertex graph with  $k \leq O((\log \log n)^{1/10})$ . In addition, if  $k$  is fixed, then the running time is  $O(n^2)$ . Furthermore, there is a polynomial time algorithm to find such a cycle if one exists.*

This algorithm has several appealing features in the context of the above discussions. First, its description and proof of correctness are much simpler and shorter than those of Robertson and Seymour's. Specifically, we can provide all the details in full in this paper. We assume that there is a function  $f(k)$  such that any graph with tree-width at least  $f(k)$  has a  $(k \times k)$ -grid minor (or a  $k$ -wall. For the definition of a  $(k \times k)$ -grid minor and a  $k$ -wall, we refer the reader to the next section.). This follows from [6,25,28,31]. The proof in [6] takes 7-8 pages. So the correctness of our proof requires at most 10 pages in total, which is much shorter than Robertson and Seymour's. Second, our running time is actually  $O(n^2)$  (for fixed  $k$ ). It is better than Robertson and Seymour's algorithm whose time complexity is  $O(n^3)$ . Third, and most importantly, our algorithm can handle even if  $k$  is a part of input, when  $k \leq O((\log \log n)^{1/10})$ . Therefore, the hidden constant is not huge.

The value " $O((\log \log n)^{1/10})$ " depends on the bound of the tree-width  $f(k)$  that forces a  $(k \times k)$ -grid minor. Currently, the best known bound is  $20^{2k^5}$  by Robertson, Seymour and Thomas [31]. But when a given graph is planar, then the situation is different. Robertson, Seymour and Thomas [31] proved that any planar graph with tree-width at least  $6k$  has a  $(k \times k)$ -grid minor. This together with our method implies the following much better result.

**Theorem 2.** *There is a polynomial time algorithm to decide whether or not there is a cycle through  $k$  independent edges on an arbitrary  $n$ -vertex planar graph with  $k \leq O((\log n / (\log \log n))^{1/4})$ . In addition, if  $k$  is fixed, then the running time is  $O(n^2)$ . Furthermore, there is a polynomial time algorithm to find such a cycle if one exists.*

Let us observe that even for planar graphs, the disjoint paths problem is NP-complete when  $k$  is as a part of input. It is likely that our problem for planar graphs is NP-complete too when  $k$  is as a part of input. Let us now give some evidence. It is well-known that a hamiltonian cycle problem in planar graphs is NP-complete. Hence, if  $k$  is as a part of input, and we are given  $k$  vertices, then it is NP-complete to decide whether or not there is a cycle through all the vertices, even for planar graphs.

We can also give a similar result for graphs on a fixed surface, i.e, bounded genus graphs. Let  $g$  be the Euler genus of the surface, and suppose  $G$  is embedded on this surface. Demaine et al. [5] proved that such a graph with tree-width at least  $6kg$  has a  $(k \times k)$ -grid minor. Using this result, we can obtain the following theorem.

**Theorem 3.** *There is a polynomial time algorithm to decide whether or not there is a cycle through  $k$  independent edges on an arbitrary  $n$ -vertex graph that is embedded on a fixed surface with the Euler genus  $g$  and with  $k \leq O((\log(n/g)/(\log \log(n/g)))^{1/4})$ . In addition, if  $k$  is fixed, then the running time is  $O(n^2)$ . Furthermore, there is a polynomial time algorithm to find such a cycle if one exists.*

Actually, we can get a better running time for Theorems [2] and [3] for fixed  $k$ , respectively. Reed et al. [27] proved that for planar graphs, there is a linear time algorithm for the disjoint paths problem for fixed  $k$ . Reed [26] extended this result to graphs on a fixed surface. Combining the technique and result in [26,27], we can actually give a linear time algorithm for Theorems [2] and [3], respectively.

So far, we are just interested in cycles through  $k$  independent edges. But our proof does imply that Theorems [1, 2] and [3] hold even if we replace  $k$  independent edges by  $k_1$  vertices and  $k_2$  independent edges, where  $k_1 + k_2 = k$ .

## 2 Overview of Our Algorithm

Our approach makes use of the ideas in Robertson-Seymour's algorithm [29], together with some apparently new algorithmic techniques. So, let us first sketch the Robertson-Seymour's algorithm on the disjoint paths problem.

At a high level, it is based on the following two cases: either a given graph  $G$  has bounded tree-width or else it has a large tree-width. In the first case, one can apply dynamic programming to a tree-decomposition of bounded tree-width, see [1,3,29]. The second case again breaks into two cases. If  $G$  has a large clique minor, then one can use this clique minor to link up a subset of the terminals in any desired way. So, suppose  $G$  does not have a huge clique minor. Then one can prove that, after deleting bounded number of vertices, there is a huge wall which is essentially planar. This makes it possible to prove that the middle vertex  $v$  of this wall is irrelevant, i.e., the disjoint paths problem is feasible in  $G$  if and only if it is in  $G - v$ . This requires the whole graph minor papers, and the structure theorem of graph minors [30]. Therefore, this part is the most difficult and complicated.

Our algorithm is based on following the same line of Robertson and Seymour's. The first step is to examine whether or not the tree-width is bounded. Again, if the tree-width is bounded, we are done. Actually, as Reed [25] pointed out, the disjoint paths problem is solvable even if  $k$  is as a part of input when the tree-width of a given graph is bounded. This is still true for our problem by using the same line of the proof given by Reed [25].



What if the tree-width is large ? It is well-known that a given graph  $G$  has a huge wall. It turns out to be possible to use a wall directly, and find an irrelevant vertex located in the huge wall. In addition, our proof just needs a  $8k^2$ -wall. There are some difficulties, since the wall is not as easy to use for our purpose as a huge clique minor. We shall show how the wall can be used to find a desired cycle through all specified independent edges. Then this idea is used to find an irrelevant vertex in the wall. The idea is inspired by Kleinberg [19]. The proof takes only a few pages. Therefore, our proof saves huge amount of spaces. In addition, our idea on looking at a wall directly improves the overall huge hidden constant, too.

Let us see our algorithm more closely. The algorithm runs as follows. First, we check whether the tree-width is small or not. If the tree-width is small, we can test (even detect it if one exists.) in linear time by the standard dynamic programming [1]. But since we also consider the case when  $k$  is as a part of input, we will use the following recent result by Perković and Reed [24]

**Theorem 4.** *The followings hold:*

1. *There exists an algorithm that, given a graph  $G$  and an integer  $m$ , finds either a subgraph of  $G$  isomorphic to an  $m$ -wall or a tree-decomposition of  $G$  of width  $w \leq 20^{2m^5}$ . The running time is  $n(m^2 + w)^{O(m^2+w)}$ .*
2. *There exists an algorithm that, given a planar graph  $G$  and an integer  $m$ , finds either a subgraph of  $G$  isomorphic to an  $m$ -wall or a tree-decomposition of  $G$  of width  $w \leq 6m$ . The running time is  $n(m^2 + w)^{O(m^2+w)}$ .*
3. *There exists an algorithm that, given a graph  $G$  on a fixed surface with the Euler genus  $g$  and an integer  $m$ , finds either a subgraph of  $G$  isomorphic to an  $m$ -wall or a tree-decomposition of  $G$  of width  $w \leq 6m/g$ . The running time is  $n(m^2 + w)^{O(m^2+w)}$ .*

Note that the tree-width bounds in the second theorem and third theorem follow from [31] and [5], respectively.

If the tree-width is small, say at most  $w$  for some constant, then as suggested by Reed [25], there is an  $O(n)k^{O(w)}$ -time algorithm to decide the  $k$  disjoint paths problem, even if  $k$  is as a part of the input. Reed’s proof [25] implies the following.

**Theorem 5.** *Suppose  $G$  has tree-width at most  $w$ . Then there is an  $O(n)k^{O(w)}$ -time algorithm to decide our problem for a cycle through specified  $k$  independent edges.*

For the tree-width large case, we only need a  $8k^2$ -wall. Hence when  $8k^2 \geq m$  in Theorem 4, then Theorems 1, 2 and 3 hold.

In the remainder of the paper, we shall discuss the large tree-width case. Suppose a  $4k^2$ -wall is given by Theorem 4. Our algorithm will find an irrelevant vertex in this wall. Then we run this algorithm recursively until the tree-width is at most  $w \leq 20^{2m^5}$  ( $w \leq 6m$  when a given graph is planar, and  $w \leq 6m/g$  when a given graph is embedded on a fixed surface with the Euler genus  $g$ ). Finally we just apply Theorem 4 for the current graph.



### 3 Definitions and Preliminaries

We will need some notation from graph minor series.

*Tree-Decomposition and Tree-width:* A *tree decomposition* of a graph  $G$  is a pair  $(T, Y)$ , where  $T$  is a tree and  $Y$  is a family  $\{Y_t \mid t \in V(T)\}$  of vertex sets  $Y_t \subseteq V(G)$ , such that the following two properties hold:

- (W1)  $\bigcup_{t \in V(T)} Y_t = V(G)$ , and every edge of  $G$  has both ends in some  $Y_t$ .
- (W2) If  $t, t', t'' \in V(T)$  and  $t'$  lies on the path in  $T$  between  $t$  and  $t''$ , then  $Y_t \cap Y_{t''} \subseteq Y_{t'}$ .

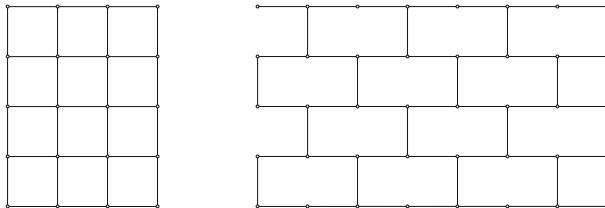
The pair  $(T, Y)$  is a *path decomposition* if  $T$  is a path. The *width* of a tree decomposition  $(T, Y)$  is  $\max_{t \in V(T)} (|Y_t| - 1)$ . Let  $(T, Y)$  be a tree decomposition of a graph  $G$ . For an edge  $tt' \in E(T)$ , let  $Z_{tt'} = Y_t \cap Y_{t'}$ . Let us recall that the *adhesion* of a tree decomposition  $(T, Y)$  is  $\max |Z_{tt'}|$  taken over all edges  $tt' \in E(T)$ .

*Grid minor and Wall:* One of the most important result concerning the tree-width is existence of grid-minor or a wall.

Let us recall that an  $r$ -*wall* is a graph which is isomorphic to a subdivision of the graph  $W_r$  with vertex set  $V(W_r) = \{(i, j) \mid 1 \leq i \leq r, 1 \leq j \leq r\}$  in which two vertices  $(i, j)$  and  $(i', j')$  are adjacent if and only if one of the following possibilities holds:

- (1)  $i' = i$  and  $j' \in \{j - 1, j + 1\}$ .
- (2)  $j' = j$  and  $i' = i + (-1)^{i+j}$ .

We can also define an  $(a \times b)$ -wall in a natural way. It is easy to see that if  $G$  has an  $(a \times b)$ -wall, then it has an  $(a \times b)$ -grid minor, and conversely, if  $G$  has an  $(a \times b)$ -grid minor, then it has an  $(a/2 \times b)$ -wall. Let us recall that the  $(a \times b)$ -grid is the Cartesian product of paths  $P_a \times P_b$ . The  $(4 \times 5)$ -grid and the  $(8 \times 5)$ -wall are shown in Figure 1.



**Fig. 1.** The  $(4 \times 5)$ -grid and the  $(8 \times 5)$ -wall

Finally, let us define canonical cycles  $C_1, \dots, C_k$  in a  $2k$ -wall  $W$ . We now delete vertices of degree 1 in  $W$ . Then  $C_k$  is an outer face boundary. Inductively, we can define  $C_i$ , which is the outer face boundary obtained by  $W$  by deleting all the cycles  $C_k, \dots, C_{i+1}$ .

## 4 Main Proof

As pointed out before, we are left with the tree-width large case. In fact, right now, a  $8k^2$ -wall is given by Theorem 4. We are trying to find an irrelevant vertex in this wall.

The following is the key lemma for our main result.

**Lemma 1.** *Let  $e_1, e_2, \dots, e_k$  be  $k$  independent edges. Let  $S = V(e_1) \cup \dots \cup V(e_k)$ . Suppose  $G$  has a  $(2k + 4)$ -wall  $W$  and let  $Q$  be the vertices of degree 3 (in  $W$ ) in the outer face boundary of  $W$ . Suppose furthermore that there are  $2k$  disjoint paths  $P_1, \dots, P_{2k}$  from  $S$  to  $Q$  such that each of these paths does not intersect any vertex in  $W$  except for the vertices in  $C_{k+2}$ , and for any three endpoints in  $Q$ , there is no 3-separations in  $W$ . Then there is a cycle  $C$  that passes through all edges  $e_1, e_2, \dots, e_k$ .*

*Proof.* We prove the statement by induction on  $k$ . When  $k = 1$ , then it is clear. Suppose  $k \geq 2$ . Take the vertex  $v_j$  in  $Q$  such that  $v_j$  is the endpoint of the path  $P_j$ . Then we assume that the other endpoint of  $P_j$  is on  $e_j$ . Let  $P_{j'} \neq P_j$  be the other path between  $e_j$  and  $Q$ . Let  $v_{j'}$  be the endpoint of  $P_{j'}$  on  $Q$ . Take the vertex in the endpoints of  $P_1, \dots, P_{2k}$  in  $Q$ , which is closest to  $v_j$  on  $C_{k+2}$ , but not  $v_{j'}$ . Call it  $v_l$ . We assume that  $v_l$  is the endpoint of the path  $P_l$ , and the other endpoint of  $P_l$  is on  $e_l$  (we also assume that  $l > j$ ). Let  $P_{l'} \neq P_l$  be the other path between  $e_l$  and  $Q$ , and  $v_{l'}$  be the endpoint of  $P_{l'}$  on  $Q$ . Let  $R$  be the path along  $C_{k+2}$  between  $v_j$  and  $v_l$ . Clearly, by our choice,  $R$  does not contain any vertex of the endpoints of  $P_1, \dots, P_{2k}$  except for  $v_j, v_l$ .

Let  $P$  be the path from  $e_j$  to  $e_l$  through  $P_j, v_j, R, v_l, P_l$ . We contract this path  $P$ ,  $e_j$  and  $e_l$  to the edge, and let  $e_{jl}$  be the resulting edge. Let  $Q'$  be the set of vertices of degree 3 (in  $W - C_{k+2}$ ) in  $C_{k+1}$ . Then by our assumption of Lemma 1, it is easy to see that there are  $2k - 2$  vertex disjoint paths from  $(S - V(e_j) - V(e_l)) \cup V(e_{jl})$  to  $Q'$  by following paths between  $C_{k+2}$  and  $C_{k+1}$  in  $W$ , together with  $2k - 2$  disjoint paths between  $(S - V(e_j) - V(e_l)) \cup V(e_{jl})$  and  $Q$ .

Now we have  $k - 1$  independent edges  $e_1, \dots, e_{j-1}, e_{j+1}, \dots, e_{l-1}, e_{l+1}, \dots, e_k, e_{jl}$  and there are  $2k - 2$  disjoint paths from  $(S - V(e_j) - V(e_l)) \cup V(e_{jl})$  to  $Q'$ . Furthermore, these paths do not intersect any vertices in  $W'$  except for the vertices on  $C_{k+1}$ . Here,  $W'$  is the  $(2k + 2)$ -wall obtained from  $W$  by deleting the outer cycle  $C_{k+2}$ . Hence the induction hypothesis is satisfied with the  $(2k + 2)$ -wall  $W'$  and  $k - 1$  independent edges  $e_1, \dots, e_{j-1}, e_{j+1}, \dots, e_{l-1}, e_{l+1}, \dots, e_k, e_{jl}$ . So by induction, there is a cycle  $C$  through all the edges  $e_1, \dots, e_{j-1}, e_{j+1}, \dots, e_{l-1}, e_{l+1}, \dots, e_k, e_{jl}$ , and clearly, this cycle can be extended to a desired cycle that passes through all edges  $e_1, e_2, \dots, e_k$ . This completes the proof.  $\square$

We are ready to prove our most important result. The proof is inspired by Kleinberg [19].

**Theorem 6.** *Let  $e_1, e_2, \dots, e_k$  be  $k$  independent edges. Let  $S = V(e_1) \cup \dots \cup V(e_k)$ . Suppose  $G$  has a  $8k^2$ -wall. If there is no separation  $(A, B)$  of order at*

most  $2k - 1$  such that  $A$  contains all the vertices in  $S$  and  $B$  contains all but at most  $k - 1$  vertices of degree 3 in  $W$ , then there is a cycle  $C$  that passes through all edges  $e_1, e_2, \dots, e_k$ .

*Proof.* By Menger’s theorem, there are  $2k$  disjoint paths from  $S$  to  $W$ . Let  $Q$  be the endvertices of these  $2k$  disjoint paths in  $W$ . There are at least  $(2k - 3)^2$  disjoint  $(2k + 3)$ -walls in  $W$  such that any of these walls does not contain any vertex in  $Q$ . We would like to apply Lemma 4. So we shall find one of  $(2k + 3)$ -subwalls, say  $W'$ , such that there are  $2k$  disjoint paths from  $S$  to  $Q'$ , any of which does not intersect any vertices in  $W_1$  except for its outer face boundary, where  $Q'$  is a vertex set of degree 3 (in  $W'$ ) in the outer cycle of  $W'$ . Then this would clearly imply the assumption of Lemma 4 by taking the  $(2k + 2)$ -wall obtained from  $W'$  by removing the outer face boundary.

We now find such a wall, using an augmenting path argument. We first consider  $2k$  disjoint paths in  $W$ , from  $Q$  to any of  $(2k + 3)$ -subwalls that does not contain any vertex of  $Q$ . If there are no such  $2k$  disjoint paths for any of them, this means that there is a separation  $(A, B)$  of order at most  $2k - 1$  in  $W$  such that  $A$  contains all the vertices in  $Q$  and  $B$  contains all but at most  $k - 1$  vertices of degree 3 in  $W$ . So, for any choice of  $Q$ , we may assume that such a separation exists. We take  $Q$  such that the order of such a separation in  $W$  is as large as possible. Subject to that, we take such a separation such that the order of  $|B|$  is as small as possible. Minimality of  $|B|$  then implies that all the vertices of  $B - A$  adjacent to  $A$  in  $W$  have degree exactly two in  $B$ , otherwise, there would be another such a separation  $(A', B')$  with  $|B'| < |B|$ . Note that we can link  $A \cap B$  to the endpoints  $Q$  in the wall to get a linkage from  $S$  to  $A \cap B$  of order  $|A \cap B|$ . By the assumptions on  $G$ , there exists an augmenting path to this linkage to get  $|A \cap B| + 1$  disjoint paths from  $S$  to the set  $B$  with endpoints  $(A \cap B) \cup x$  for some  $x$  in  $B - A$ . Let us keep this in mind.

Since there are no such separations in  $G$ , there must be a path  $P_1$  between  $A - B$  and  $B - A$  in  $G$  (We call it *augmenting path*). Let  $x_1$  be the endvertex of  $P_1$  in  $B - A$ . By the minimality of  $B$  and the above remark, there are  $|A \cap B| + 1$  disjoint paths from  $Q$  to  $(A \cap B) \cup \{x_1\}$  in  $W \cup P_1$ . We call this procedure *augmenting process*. Let  $W_1 = W \cup P_1$ . If there is a separation  $(A', B')$  of order at most  $2k - 1$  in  $W_1$  such that  $A'$  contains all the vertices in  $Q$  and  $B'$  contains all but at most  $k - 1$  vertices of degree 3 in  $W$ , then, again, we take such a separation  $(A', B')$  such that the order of  $|B'|$  is as small as possible. Clearly  $|A' \cap B'| > |A \cap B|$ . Since there are no such separations in  $G$ , there must be a path  $P_2$  between  $A' - B'$  and  $B' - A'$ . Let  $x_2$  be the endvertex of  $P_2$  in  $B' - A'$ . We take such a path  $P_2$  so that there are  $|A' \cap B'| + 1$  disjoint paths from  $Q$  to  $(A' \cap B') \cup \{x_2\}$  in  $W_1 \cup P_2$  (by rechoosing the path  $P_1$  (but the same endpoints), if necessary, such a path exists because of our assumption of connectivity in Theorem 6). Let  $W_2 = W_1 \cup P_2$ . We now repeat this process until there are no such separations of order at most  $2k - 1$ . Since, in each step, the order of the separation  $|A' \cap B'|$  increases by at least 1, hence this implies that this augmenting process stops when we perform at most  $2k - 1$  times. Let us observe that even if we add at most  $2k - 1$  disjoint paths to  $W$ , there are

at least  $2k^2$  disjoint  $(2k + 6)$ -subwalls, any of which contains neither a vertex in  $Q$  nor a vertex in the endpoints of these paths. Therefore, if there are no such separations of order at most  $2k - 1$ , clearly there are  $2k$  disjoint paths from  $Q$  to the vertex set  $Q'$  of a  $(2k + 6)$ -subwall  $W'$  such that any of these  $2k$  disjoint paths does not intersect any vertex in  $W'$ , except for its outer face boundary, where  $Q'$  is a vertex set of degree 3 (in  $W'$ ) in the outer face boundary of  $W'$ . We now claim that there are  $2k$  disjoint paths from  $S$  to  $Q'$  such that any of these  $2k$  disjoint paths does not intersect any vertex in  $W'$ , except for its outer face boundary. Our choice of  $Q$  (maximality of the order of the separation) implies that for each iteration of the augmenting step with a separation  $(A', B')$ , there are  $|A' \cap B'|$  disjoint paths from  $S$  to  $A' \cap B'$  such that any of these paths does not use any vertex in  $B' - A'$ . It follows that there are  $2k$  disjoint desired paths from  $S$  to  $Q'$  such that none of them uses a vertex in  $W'$ , except for the vertices on the outer face boundary of  $W'$ . Let  $W''$  be the  $(2k + 4)$ -wall obtained from  $W'$  by removing the outer cycle of  $W'$ . By applying Lemma 11 to  $S$  and  $W''$ , there is a desired cycle. This completes the proof.  $\square$

Having proved Theorem 6, our algorithm is easy. We can prove the following.

**Theorem 7.** *Suppose  $G$  has a  $(8k + 3)^2$ -wall  $W$ . Then there is an  $O(n)$  algorithm to find an irrelevant vertex  $v$  in  $W$ , i.e., there is a desired cycle through all the edges  $e_1, e_2, \dots, e_k$  in  $G$ , if and only if  $G - v$  has.*

*Proof.* Let  $S = V(e_1) \cup \dots \cup V(e_k)$ . Since we only need a  $8k^2$ -wall in the proof of Theorem 6 and  $G - v$  has a  $8k^2$ -wall  $W'$  for any vertex  $v$  of degree 3 in  $W$ , so this result would follow if there would be a vertex  $v$  of degree 3 in  $W$  such that there would be no separation  $(A, B)$  of order less than  $2k - 1$  in  $G - v$  so that  $A$  contains all the vertices in  $S$  and  $B$  contains all but at most  $k - 1$  vertices of degree 3 in  $W$ . In fact, such a vertex  $v$  would be irrelevant by using Theorem 6. So suppose such a separation of order at most  $2k - 1$  exists for each vertex  $v$  of degree 3 in  $W$ .

In order to find an irrelevant vertex in  $W$ , we first detect a minimum separation  $(A, B)$  such that  $A$  contains all the vertices in  $S$ , and  $B$  contains all but at most  $k - 1$  vertices of degree 3 in the wall  $W$ . Subject to this, we take such a separation such that  $|B|$  is as small as possible. Such a separation can be found by the result in [12]. The running time is  $O(f(k)n)$ , where  $f(k)$  depends on the number of vertices of degree 3 in the wall  $W$ , which is at most  $(8k + 3)^2$ . Note that  $B$  has a  $(8k^2 + 1)$ -wall  $W'$ . We claim that any vertex  $v$  of degree 3 in  $W'$  in  $B - A$  is irrelevant. Clearly, we can reduce the problem in  $G$  to that in  $B$  with at most  $2k - 1$  terminals in  $A \cap B$ . Since the minimality of  $|B|$  implies that any vertex of  $v$  in  $B - A$  does not create any separation of smaller order, and  $G - v$  has a  $4k^2$ -wall  $W''$ , therefore, by Theorem 6, this problem is feasible in  $B$  with respect to the terminals in  $A \cap B$  if and only if it is in  $B - v$ . So,  $v$  is irrelevant in  $B$ , and hence in  $G$ . This completes the proof.  $\square$

Theorem 7 implies our algorithm. When the tree-width is small, we can decide the problem, and actually, detect a desired cycle if one exists, as discussed before. If the tree-width is large, then we first apply Theorem 4 to get a  $(8k + 3)^2$ -wall.

Then we apply Theorem 7 to find an irrelevant vertex  $v$ . Then we run our algorithm to  $G - v$  recursively until the current graph has bounded tree-width. Hence the running time is  $O(n^2)$ .

## Acknowledgement

We would like to thank the referee for suggesting a fix of a hole in the original version.

## References

1. Arnborg, S., Proskurowski, A.: Linear time algorithms for NP-hard problems restricted to partial  $k$ -trees. *Discrete Appl. Math.* 23, 11–24 (1989)
2. Berge, C.: *Graphs and Hypergraphs*. North-Holland, Amsterdam (1973)
3. Bodlaender, H.L.: A linear-time algorithm for finding tree-decomposition of small treewidth. *SIAM J. Comput.* 25, 1305–1317 (1996)
4. Bondy, J.A., Lovász, L.: Cycles through specified vertices of a graph. *Combinatorica* 1, 117–140 (1981)
5. Demaine, E.D., Fomin, F., Hajiaghayi, M., Thilikos, D.: Subexponential parameterized algorithms on bounded-genus graphs and  $H$ -minor-free graphs. *J. ACM* 52, 1–29 (2005)
6. Diestel, R., Gorbunov, K.Y., Jensen, T.R., Thomassen, C.: Highly connected sets and the excluded grid theorem. *J. Combin. Theory Ser. B* 75, 61–73 (1999)
7. Dirac, G.A.: In abstrakten Graphen vorhandene vollständige 4-Graphen und ihre unterteilungen. *Math. Nachr.* 22, 61–85 (1960)
8. Erdős, P.L., Györi, E.: Any four independent edges of a 4-connected graph are contained in a circuit. *Acta Math. Hungar.* 46, 311–313 (1985)
9. Gabow, H.: Finding paths and cycles of superpolylogarithmic length. In: *STOC 2004*, Chicago, Illinois, USA, pp. 407–416 (2004)
10. Häggkvist, R., Thomassen, C.: Circuits through specified edges. *Discrete Math.* 41, 29–34 (1982)
11. Holton, D.A., McKay, B.D., Plummer, M.D., Thomassen, C.: A nine point theorem for 3-connected graphs. *Combinatorica* 2, 53–62 (1982)
12. Ibaraki, T., Nagamichi, H.: A linear time algorithm for finding a sparse  $k$ -connected spanning subgraph of a  $k$ -connected graph. *Algorithmica* 7, 583–596 (1992)
13. Johnson, D.: The Many Faces of Polynomial Time. *J. Algorithms* 8, 285–303 (1987)
14. Kawarabayashi, K.: One or two disjoint circuits cover independent edges, Lovász-Woodall Conjecture. *J. Combin. Theory Ser. B* 84, 1–44 (2002)
15. Kawarabayashi, K.: Two circuits through independent edges (manuscript, 1999)
16. Kawarabayashi, K.: An extremal problem for two circuits through independent edges (manuscript, 1999)
17. Kawarabayashi, K.: Proof of Lovász-Woodall Conjecture (in preparation)
18. Kawarabayashi, K.: Cycles through prescribed vertex set in  $N$ -connected graphs. *J. Combin. Theory Ser. B* 90, 315–323 (2004)
19. Kleinberg, J.: Decision algorithms for unsplittable flows and the half-disjoint paths problem. In: *Proc. 30th ACM Symposium on Theory of Computing*, pp. 530–539 (1998)

20. LaPaugh, A.S., Rivest, R.L.: The subgraph homomorphism problem. *J. Comput. Sys. Sci.* 20, 133–149 (1980)
21. Lomonosov, M.V.: Cycles through prescribed elements in a graph. In: Korte, Lovász, Prömel, Schrijver (eds.) *Paths, Flows, and VLSI Layout*, pp. 215–234. Springer, Berlin (1990)
22. Lovász, L.: Problem 5. *Period. Math. Hungar.* 82 (1974)
23. Lovász, L.: Exercise 6.67. In: *Combinatorial Problems and Exercises*. North-Holland, Amsterdam (1979)
24. Perkovic, L., Reed, B.: An improved algorithm for finding tree decompositions of small width. *International Journal on the Foundations of Computing Science* 11, 81–85 (2000)
25. Reed, B.: Tree width and tangles: a new connectivity measure and some applications. In: *Surveys in Combinatorics, London*. London Math. Soc. Lecture Note Ser., vol. 241, pp. 87–162. Cambridge Univ. Press, Cambridge (1997)
26. Reed, B.: Rooted Routing in the Plane. *Discrete Applied Mathematics* 57, 213–227 (1995)
27. Reed, B., Robertson, N., Schrijver, A., Seymour, P.D.: Finding disjoint trees in planar graphs in linear time. In: *Graph structure theory (Seattle, WA, 1991)* *Contemp. Math.*, pp. 295–301. Amer. Math. Soc., Providence (1993)
28. Robertson, N., Seymour, P.D.: Graph minors. V. Excluding a planar graph. *J. Combin. Theory Ser. B* 41, 92–114 (1986)
29. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory Ser. B* 63, 65–110 (1995)
30. Robertson, N., Seymour, P.D.: Graph minors. XVI. Excluding a non-planar graph. *J. Combin. Theory Ser. B* 89, 43–76 (2003)
31. Robertson, N., Seymour, P.D., Thomas, R.: Quickly excluding a planar graph. *J. Combin. Theory Ser. B* 62, 323–348 (1994)
32. Sanders, D.P.: On circuits through five edges. *Discrete Math.* 159, 199–215 (1996)
33. Thomassen, C.: Note on circuits containing specified edges. *J. Combin. Theory Ser. B* 22, 279–280 (1977)
34. Woodall, D.R.: Circuits containing specified edges. *J. Combin. Theory Ser. B* 22, 274–278 (1977)

# The Stable Roommates Problem with Choice Functions

Tamás Fleiner\*

Budapest University of Technology and Economics,  
Department of Computer Science and Information Theory,  
Magyar tudósok körútja 2. H-1117, Budapest, Hungary  
fleiner@cs.bme.hu

**Abstract.** The stable marriage theorem of Gale and Shapley states that for  $n$  men and  $n$  women there always exists a stable marriage scheme, that is, a set of marriages such that no man and woman exists that mutually prefer one another to their partners. The stable marriage theorem was generalized in two directions: the stable roommates problem is the “one-sided” version, where any two agents on the market can form a partnership. The generalization by Kelso and Crawford is in the “two-sided” model, but on one side of the market agents have a so-called substitutable choice function, and stability is interpreted in a natural way. It turned out that even if both sides of the market have these substitutable choice functions, there still exists a stable assignment. This latter version contains the “many-to-many” model where up to a personal quota, polygamy is allowed for both men and women in the two-sided market.

The goal of this work is to solve the stable partnership problem, a generalization of the one-sided model with substitutable choice functions. We do not quite reach that: besides substitutability, we also need the increasing property for the result. Luckily, choice functions in well-known stable matching theorems comply with this property. The main result is a generalization of Irving’s algorithm, that is the first efficient method to solve the stable roommates problem. This general algorithm allows us to deduce a generalization of Tan’s result on characterizing the existence of a stable matching and to prove a generalization of the so-called splitting property of many-to-many stable matchings. We show that our algorithm is linear-time in some sense and indicate that for general (i.e. not necessary increasing) substitutable choice functions the stable partnership problem is NP-complete.

## 1 Introduction

Gale and Shapley [9] introduced their famous marriage model almost a half century ago. The model consists of  $n$  men and  $n$  women such that each person has

---

\* Research is supported by OTKA grants K69027 and K60802, the MTA-ELTE Egerváry Research Group (EGRES) and the János Bolyai research fellowship of the Hungarian Academy of Sciences.

a linear preference order on the members of the opposite gender. The marriage theorem states, that for each such model there exists a stable marriage scheme, that is, a set of disjoint couples in such a way that no man and woman mutually prefer one another to their partners. It turned out that variants of the model are useful in Game Theory, Economics, Graph Theory, Complexity Theory, Combinatorial Optimization and the Theory of Algorithms. Also, stable matchings have a rich structure, and this also led to further generalizations. There are two natural directions of these generalizations. For the first one, we drop the two-sided property of the market. This way we get the stable roommates problem: any two agents may have a relationship, the solution of the problem is more difficult, and a stable matching does not always exist. Irving [11] was the first who gave an efficient algorithm that finds a stable matching in a given model if it exists. Since then, many different approaches are known for the same problem. (Cf. Tan [17], Tan and Hsueh [18], Subramanian [16], Feder [5] and others. See also the book of Roth and Sotomayor [13] and of Gusfield and Irving [10] for further details).

The other direction for a possible generalization is that we allow that an agent may participate in several relationships, so instead of a matching, we look for a certain subgraph of the underlying graph with degree prescriptions. This is done by keeping linear preference orders, but introducing quotas for the agents (this model is present already in the original paper of Gale and Shapley), or in a much more general way, by choice functions. This is what was initiated by Crawford and Knoer [4], and continued by Kelso and Crawford [12]. This choice function model is closely related to the lattice theoretical fixed point theorem of Knaster and Tarski [19] as it was pointed out by Fleiner [6]. It is worth mentioning that the nonbipartite stable roommates problem has to do with an other fixed point theorem: as Aharoni and Fleiner [1] observed, it is a special case of the well known game theoretical Scarf's lemma [14]. (Scarf's lemma can be regarded as a discrete version of Kakutani's fixed point theorem, which is a generalization of the topological fixed point theorem of Brouwer).

In the present work, we move into both directions. In Section 2, we describe our one-sided (nonbipartite) model and define the stability concept by choice functions. By a generalization of Irving's algorithm, we solve the defined stable partnership problem in Section 3 for so called increasing choice functions. Furthermore, the same way as Tan [17] did, we show that a certain half-integral solution always exists, and such a solution is either a generalized stable matching or it is an immediate proof for the nonexistence of it. The reader who finds it difficult to follow all the details may focus only on the solution of the stable partnership problem and ignore the existence of half-integral solutions. It is fairly easy to reduce the algorithm to the integral stable partnership problem, and proofs become much easier. If the choice functions of the model are given by an oracle then the algorithm works in polynomial time, and Section 4 gives a detailed analysis. We also show there that for general choice functions the stable partnership problem is NP-complete. Section 5 contains a structural result on stable partnerships: we generalize a result of Fleiner [8] (that was independently



found by Teo et al. [15]) on the splitting property of stable matchings. We conclude in Section 6 with raising the question of a possible generalization of the well-known Scarf’s lemma. Some proofs are missing from this present work due to space limits. The interested reader can find them in [7].

## 2 Preliminaries

Let  $G = (V, E)$  be a finite graph. For a subset  $E'$  of  $E$  and vertex  $v$  of  $G$  let us denote by  $E'(v)$  the set of edges of  $E'$  that are incident with  $v$ . Let  $C_v$  denote the *choice function of vertex  $v$* , i.e. function  $C_v : 2^{E(v)} \rightarrow 2^{E(v)}$  maps any set  $X$  of edges incident with  $v$  to a subset of  $X$  that  $v$  chooses from  $X$ . We shall assume that choice functions we handle are *substitutable*, that is, if  $x \neq y$  and  $x \in C_v(X)$ , then  $x \in C_v(X \setminus \{y\})$  holds. This roughly means that if agent  $v$  would select some option  $x$  from set  $X$  of available options, then  $v$  would still select option  $x$  even if some other options are not available any more. For choice function  $C_v$ , let us define function  $\overline{C}_v$  of ignored options by  $\overline{C}_v(X) := X \setminus C_v(X)$ . It is useful to see a connection between the above two notions.

**Theorem 1.** *A choice function  $C$  on a finite groundset is substitutable if and only if  $\overline{C}$  is monotone, that is, if  $Y \subseteq X$  implies that  $\overline{C}(Y) \subseteq \overline{C}(X)$ .*

Choice functions  $C$  with the property that  $\overline{C}$  is monotone are called *comonotone* in [6]. So by theorem 1, a choice function on a finite ground set is comonotone if and only if it is substitutable. Note that on infinite ground sets, comonotonicity of a choice function is a stronger property than substitutability.

*Proof.* For finite groundsets, substitutability is equivalent with the property that for any  $Y \subseteq X$  we have  $Y \cap C(X) \subseteq C(Y)$ . Monotonicity of  $\overline{C}$  is equivalent with the property that for any  $Y \subseteq X$ ,  $C(X)$  is disjoint from  $\overline{C}(Y)$ , which is clearly equivalent with the first property.  $\square$

We say that a subset  $S$  of  $E$  is a *stable partnership* if for any vertex  $v$  we have  $C_v(S(v)) = S(v)$  (this property is often called *individual rationality*: no agent  $v$  will participate in a partnership  $vx$  if  $vx$  is an ignored option for  $v$  when all actual partnerships of  $v$  are offered) and no *blocking edge*  $e = uv \notin S$  exists such that both  $e \in C_u(S(u) \cup \{e\})$  and  $e \in C_v(S(v) \cup \{e\})$  holds. This corresponds to the situation that besides their actual partnerships, both  $u$  and  $v$  are interested in forming partnership  $uv$ . The *stable partnership problem* is given by a graph  $G = (V, E)$  and choice functions  $C_v$  for each vertex. Our task is to decide whether a stable partnership exists, and if yes, we have to find one.

An example of a substitutable choice function  $C_v$  on ground-set  $E(v)$  is a *linear choice function*: we have a linear order on  $E(v)$  and  $C_v(X)$  is the minimal element of  $X$  according to this linear order. If for all vertices  $v$  of  $G$ , choice function  $C_v$  is linear then a stable partnership is simply a stable matching. Another possible choice function is when each vertex  $v$  has a “quota”  $b(v)$ , and  $C_v(X)$  is the  $b(v)$  smallest elements of  $X$  according to the linear order. For these

linear choice functions with quotas, a stable partnership is nothing but the well-studied “many-to-many” stable matching (or stable  $b$ -matching). An even more general substitutable choice function can be defined with matroids: let  $\mathcal{M}$  be a matroid on  $E(v)$  and fix a linear order on  $E(v)$  as well. Let  $C_v(F)$  be the output of the greedy algorithm on  $F$ , that is, we scan the elements of  $F$  in the given linear order, and scanned element  $e$  is selected into  $C_v(F)$ , if  $e$  together with the previously selected elements is independent in  $\mathcal{M}$ .

Fleiner [6] generalized a result of Kelso and Crawford [12] by showing the following theorem.

**Theorem 2 (Fleiner [6]).** *If  $G = (V, E)$  is a finite bipartite graph and for each vertex  $v$ , choice function  $C_v$  on  $E(v)$  is substitutable, then there always exists a stable partnership.* □

Choice functions in the above examples satisfy the following additional property. Choice function  $C_v$  is *increasing* if  $Y \subseteq X$  implies that  $|C_v(Y)| \leq |C_v(X)|$ . This roughly means that if extra choices are added, then  $v$  selects at least as many options as  $v$  would have picked without the extra ones. For a choice function  $C_v$ , we say that subset  $X$  of  $E$   $v$ -dominates element  $x$  of  $E(v)$  if  $x \in \overline{C}_v(X(v) \cup \{x\})$ . Less formally,  $x$  is dominated, if  $v$  is not interested in  $x$  even if beside the set  $X$  of possible options we make option  $x$  available for  $v$ . If it causes no ambiguity, then instead of  $v$ -domination we may speak about *domination*. For a choice function  $C_v$ , let  $D_v(X)$  denote the set of elements ( $v$ -)dominated by  $X$ . Clearly,  $\overline{C}_v(X) \subseteq D_v(X)$  and  $C_v(X) = X(v) \setminus D_v(X)$  holds for any subset  $X$  of  $E(v)$ .

**Lemma 1.** *If choice function  $C_v$  is substitutable then dominance function  $D_v$  is monotone. If choice function  $C_v$  is substitutable and increasing and  $Y \subseteq D_v(X)$  then  $C_v(X \cup Y) = C_v(X)$ , thus  $D_v(X \cup Y) = D_v(X)$ .*

*Proof.* Assume first that  $A \subseteq B$  and  $a \in D_v(A)$ . Monotonicity of  $\overline{C}_v$  implies that  $a \in \overline{C}_v(A \cup \{a\}) \subseteq \overline{C}_v(B \cup \{a\})$ , so  $a \in D_v(B)$ , hence  $D_v$  is monotone. (Actually, the implication is true in the other direction, as well. If  $D_v$  is monotone then  $X \mapsto X \cap D_v(X) = \overline{C}_v(X)$  is also also monotone, hence  $C_v$  is substitutable by Lemma [1].)

For the second part, let  $y \in Y \cup \overline{C}_v(X)$  be an arbitrary element dominated by  $X$ . So  $y \in \overline{C}_v(X \cup \{y\}) \subseteq \overline{C}_v(X \cup Y)$ , where the second relation follows from the monotonicity of  $\overline{C}_v$ . Hence  $Y \cup \overline{C}_v(X) \subseteq \overline{C}_v(X \cup Y)$ , that is,  $C_v(X \cup Y) = (X \cup Y) \setminus \overline{C}_v(X \cup Y) \subseteq (X \cup Y) \setminus (\overline{C}_v(X) \cup Y) \subseteq X \setminus \overline{C}_v(X) = C_v(X)$ . As  $X \subseteq X \cup Y$ , the increasing property of  $C_v$  implies that  $|C_v(X)| \leq |C_v(X \cup Y)|$ , so  $C_v(X) = C_v(X \cup Y)$  follows. □

The notion of dominance allows us to reformulate the notion of a stable partnership.

**Theorem 3.** *If  $G = (V, E)$  is a finite graph and for each vertex  $v$ , choice function  $C_v$  on  $E(v)$  is substitutable then  $S$  is a stable partnership if and only if  $E \setminus S = \bigcup_{v \in V} D_v(S(v))$ , that is, if  $S$  dominates exactly  $E \setminus S$ .*

*Proof.* If  $S$  is a stable partnership then by individual rationality, no edge of  $S$  is dominated by  $S$ . As no blocking edge exists, each edge outside  $S$  is dominated by  $S$ . On the other hand, if  $E \setminus S = \bigcup_{v \in V} D_v(S \cap E(v))$  then no edge of  $S$  is dominated by  $S$ , thus  $S$  is individually rational. As each edge outside  $S$  is dominated, no blocking edge exists.  $\square$

Let  $C_v$  be a choice function and let  $X \subseteq E(v)$ . For an edge  $x$  in  $C_v(X)$  the  $X$ -replacement of  $x$  according to  $C_v$  is the set  $R = C_v(X \setminus \{x\}) \setminus C_v(X)$ . Roughly speaking, if option  $x$  is not available for  $v$  any more, then  $v$  selects from  $X$  options of  $R$  instead of  $x$ .

**Lemma 2.** *If  $C_v$  is an increasing and substitutable choice function on  $E(v)$ ,  $X \subseteq E(v)$  and  $x \in C_v(X)$ , then the  $X$ -replacement  $R$  of  $x$  contains at most one element.*

*Proof.* We have  $C_v(X) \setminus \{x\} \subseteq C_v(X \setminus \{x\})$  by substitutability, so  $C_v(X \setminus \{x\}) = C_v(X) \cup R \setminus \{x\}$ . From the increasing property of  $C_v$ , we get  $|C_v(X)| \geq |C_v(X \setminus \{x\})| = |C_v(X) \cup R \setminus \{x\}| = |C_v(X)| + |R| - 1$ , and the lemma follows.  $\square$

Let  $G = (V, E)$  be a graph and for each vertex  $v$ , let  $C_v$  be a choice function on  $E(v)$ . Let  $S$  be a subset of  $E$  and fix disjoint subsets  $S_1, S_2, \dots, S_l$  of  $S$  such that each  $S_i$  is an odd cycle (i.e. a closed walk in  $G$ ) with a fixed orientation. (Note that cycle  $S_i$  is not necessarily a circuit, as  $S_i$  can traverse the same vertex several times.) Let  $S_i^+(v)$  and  $S_i^-(v)$  denote the set of edges of  $S_i$  that leave and enter vertex  $v$ , respectively. Define edge set  $S^+(v) := S(v) \setminus \bigcup_{i=1}^l S_i^-(v)$  as the unoriented edges of  $S(v)$  together with the arcs of the  $S_i$ 's that leave  $v$ . Similarly let  $S^-(v) := S(v) \setminus \bigcup_{i=1}^l S_i^+(v)$ , denote the set unoriented edges of  $S(v)$  and all arcs of the  $S_i$ 's that enter  $v$ . We say that  $(S, S_1, \dots, S_l)$  is a *stable half-partnership* if

1. for each  $v \in V$ ,  $C_v(S(v)) = S^+(v)$ . Moreover,
2. If arcs  $e \in S_i^-(v)$  and  $f \in S_i^+(v)$  are consecutive on  $S_i$  then  $\{e\}$  is the  $S(v)$ -replacement of  $f$  according to  $C_v$ .
3.  $E \setminus S = \bigcup_{v \in V} D_v(S^-(v))$ .

A consequence of the definition is that if  $(S, S_1, \dots, S_l)$  is a stable half-partnership and  $e = uv$  is an edge then exactly one of the following three possibilities holds. Either  $e$  is an unoriented edge of  $S$  (that does not belong to any of the  $S_i$ 's), or  $e$  is an edge of some  $S_i$ , and hence if  $e = S_i^-(v)$  then  $e$  is dominated by  $S_i^+(v)$ , or  $e \notin S$  and hence  $e$  is dominated by  $S^-(u)$  at some vertex  $u$  of  $e$ . If  $(S, S_1, \dots, S_l)$  has properties **1** and **2**, and edge  $e = uv \notin S$  is not dominated (i.e.  $e = uv$  and  $e \notin D_u(S^-(u)) \cup D_v(S^-(v))$ ) then we say that  $e$  is *blocking*  $(S, S_1, \dots, S_l)$ . Observe that if  $(S)$  is a stable half-partnership (that is, no oriented odd cycles are present) if and only if  $S$  is a stable partnership. Now we can state our main result.

**Theorem 4.** *If  $G = (V, E)$  is a finite graph and for each vertex  $v$ , choice function  $C_v$  on  $E(v)$  is increasing and substitutable then there exists a stable half-partnership. Moreover, if  $(S, S_1, \dots, S_l)$  and  $(S', S'_1, \dots, S'_m)$  are stable half-partnerships, then  $l = m$  and sets of oriented cycles  $\{S_1, \dots, S_l\}$  and  $\{S'_1, \dots, S'_m\}$  are identical.*

**Corollary 1.** *If  $(S, S_1, \dots, S_l)$  is a stable half-partnership then either  $l = 0$  and  $S$  is a stable partnership, or no stable partnership exists whatsoever.  $\square$*

Corollary 1 shows that to solve the stable partnership problem in case of increasing substitutable choice functions, it is enough to find a stable half-partnership. Note that Theorem 4 is a generalization of Tan's result [17] on stable half-matchings (or on "stable partitions" in Tan's terminology).

### 3 Proof of the Main Result

To prove our main result, we follow Tan's method. Tan extended Irving's algorithm such that it finds a stable half-matching, and, with the help of the algorithm, he proved the unicity of the oriented odd cycles. Here, instead of linear orders, we work with increasing substitutable choice functions. To handle this situation, we shall generalize Irving's algorithm to our setting. Irving's algorithm works in such a way that it keeps on deleting edges such that no new stable matching is created after a deletion, and, if there was a stable matching before a deletion, there should be one after it, as well. Irving's algorithm terminates if the actual graph is a matching, which, by the deletion rules is a stable matching for the original problem. Similarly, our algorithm will delete edges in such a way that after a deletion no new stable half-partnership can be created. Moreover, if there was a stable half-partnership  $(S, S_1, \dots, S_l)$  before some deletion, then we cannot delete an edge of any of the  $S_i$ 's and at least one stable half-partnership has to survive the deletion. If our algorithm terminates then we are left with a graph  $G'$  such that edge set  $E(G')$  of  $G'$  is a stable half partnership of  $G'$ , hence it is a stable half partnership of  $G$ , as well. Our algorithm has different deletion rules, and there is a priority of them. The algorithm always takes a highest priority step that can be made. In this section we describe and justify our algorithm. Note that our definitions and theorems always refer to the "actual" graph, so if one checks how the algorithm works on a given instance then the problem is changing after each step. In particular, the set  $E$  of edges is changing, as it is the edge set of the actual graph.

To start the algorithm we need some definitions. We say that the *first choices* of  $v$  are the edges of  $C_v(E(v))$ . These are the best possible options for agent  $v$ . If edge  $e = vw$  is a first choice of  $v$  then we call arc  $e = vw$  a 1-arc. Note that if  $vw$  is a 1-arc then it is possible that  $wv$  is also a 1-arc. Let  $A$  denote the set of 1-arcs. For a vertex  $v$  let  $A^+(v)$  and  $A^-(v)$  stand for the set of 1-arcs that are oriented away from  $v$  and towards  $v$ , respectively.

The **1st priority (proposal) step** is that we find and orient all 1-arcs.

As the problem did not change (we did not delete anything), the set of stable half-partnerships is the same as it was before the orientation. After we found all 1-arcs, we execute the

**2nd priority (rejection) step:** If  $D_v(A^-(v)) \neq \emptyset$  for some vertex  $v$  then we delete  $D_v(A^-(v))$ .

**Lemma 3.** *The set of stable half-partnerships does not change by a 2nd priority step.*

*Proof.* Assume that  $(S, S_1, \dots, S_l)$  is a stable half-partnership after the deletion. This means that for each 1-arc  $f = uv$  of  $A^-(v)$  either  $f$  belongs to  $S^-(v)$ , or, as  $f$  cannot be dominated by  $S^-(u)$  at  $u$ ,  $f$  is dominated by  $S^-(v)$  at  $v$ . Lemma □ implies that  $D_v(A^-(v)) = D_v(C_v(A^-(v))) \subseteq D_v(S^-(v) \cup C_v(A^-(v))) = D_v(S^-(v))$ , hence no deleted edge can block  $(S, S_1, \dots, S_l)$ . This means that no new stable half-partnership can emerge after a 2nd priority deletion.

Assume now that  $(S, S_1, \dots, S_l)$  is a stable half-partnership before the deletion and some edge  $e \in D_v(A^-(v))$  belongs to  $S$ . Similarly to the previous argument, this means that for each 1-arc  $f = uv$  of  $C_v(A^-(v))$  either  $f$  belongs to  $S^-(v)$ , or (as  $f$  cannot be dominated by  $S^-(u)$  at  $u$ )  $f$  is dominated by  $S^-(v)$  at  $v$ . Lemma □ implies that  $e \in D_v(A^-(v)) = C_v(D_v(A^-(v))) \subseteq D_v((S^-(v) \cup C_v(A^-(v))) = D_v(S^-(v))$ , so  $e$  cannot belong to  $S(v)$ , a contradiction. □

Later we need the following lemma.

**Lemma 4.** *If no 1st and 2nd priority steps can be made then  $|A^+(v)| = |A^-(v)|$  for each vertex  $v$ .*

*Proof.* By the increasing property of  $C_v$ , we have  $|A^-(v)| = |C_v(A^-(v))| \leq |C_v(E(v))| = |A^+(v)|$ . So each vertex  $v$  has at least as many outgoing 1-arcs as the number of 1-arcs entering  $v$ . As both the total number of outgoing 1-arcs and the total number of ingoing 1-arcs is exactly  $|A|$ , the previous inequality must be an equality for each vertex  $v$ . □

If no more 1st and 2nd priority steps can be made, we check for replacements. For a 1-arc  $e = uv$ , let  $e^r = uw$  denote the  $E(u)$ -replacement of  $e$  according to  $C_u$ . (It might happen that  $e^r$  does not exist).

**3rd priority (replacement) step:** For each 1-arc  $e = uv$ , find  $E(u)$ -replacement  $e^r$ .

As we do not delete anything in a 3rd priority step, the set of stable partnerships does not change by this step. Next we study replacements of 1-arcs.

**Lemma 5.** *Assume that no 1st and 2nd priority steps can be made and that 1-arc  $e = uv$  is not bidirected, that is,  $vu$  is not a 1-arc. Then the  $E(u)$ -replacement  $e^r$  of  $e$  is a unique edge of  $E(u)$ .*

*Proof.* By Lemma □ and the increasing property of  $C_u$ , we have  $|A^+(u)| = |A^-(u)| = |C_u(A^-(u))| \leq |C_u(E(u) \setminus \{e\})| \leq |C_u(E(u))| = |A^+(u)|$ , hence there

is equality throughout. In particular, we see that  $|C_u(E(u) \setminus \{e\})| = |A^+(u)|$ . By substitutability of  $C_u$ , we have  $A^+(u) \setminus \{e\} = C_u(E(u)) \setminus \{e\} \subseteq C_u(E(u) \setminus \{e\})$ . This means that the  $E(u)$ -replacement of  $e$  (that is,  $C_u(E(u) \setminus \{e\}) \setminus A^+(u)$ ) is a unique edge of  $E(u)$ .  $\square$

**Lemma 6.** *Assume that no 1st and 2nd priority step can be executed and  $e = uv$  is a 1-arc such that  $vu$  is not a 1-arc. If  $e^r = uw$  is the  $E(u)$ -replacement of  $e$ , then  $D_w(\{e^r\} \cup A^-(w))$  contains exactly one 1-arc  $e_r^r = xw$ . Moreover, 1-arc  $e_r^r = xw$  has the property that its inverse  $wx$  is not a 1-arc.*

*Proof.* By the 2nd priority step  $e^r \notin D_w(A^-(w))$ , hence  $e^r \in C_w(\{e^r\} \cup A^-(w))$ . The increasing property of  $C_w$  gives that  $|A^-(w)| = |C_w(A^-(w))| \leq |C_w(\{e^r\} \cup A^-(w))| \leq |C_w(E(w))| = |A^+(w)| = |A^-(w)|$ , where the last equality is due to Lemma 4. So we have equality throughout, i.e.  $|A^-(w)| = |C_w(\{e_r^r\} \cup A^-(w))|$ , so  $e^r$  has a unique  $\{e^r\} \cup A^-(w)$ -replacement  $e_r^r = xw$ . Clearly, if  $wx$  was a 1-arc then  $e_r^r \in C_w(\{e_r^r\} \cup A^-(w))$  holds, a contradiction.  $\square$

Assume now that no 1st, 2nd and 3rd priority steps are possible and consider the following two cases.

**Case 1.** All 1-arcs are bidirected, that is, if  $e = uv$  is a 1-arc, then its opposite  $vu$  is also a 1-arc. In other words,  $A^+(v) = A^-(v) = A(v) = E(v)$  for each vertex  $v$ . This means that the edge set of our graph is a stable partnership, the algorithm terminates and outputs  $(S)$ .

**Case 2.** There exists a 1-arc  $e = uv$  that is not bidirected. So there is an  $E(u)$ -replacement  $e^r$  of  $e$ . Edge  $e_r^r$  in Lemma 6 is another 1-arc that is not bidirected. Following the alternating sequence of nonbidirected 1-arcs and their replacements, we shall find a sequence  $(e_1, (e_1)^r, e_2, (e_2)^r, \dots, e_m, (e_m)^r, e_{m+1} = e_1)$  in such a way that  $(e_i)^r = e_{i+1}$  for  $i = 1, 2, \dots, m$  and edges  $e_1, e_2, \dots, e_m$  are different 1-arcs, none of them is bidirected. After Irving, we call such an alternating sequence  $(e_1, (e_1)^r, e_2, (e_2)^r, \dots, e_m, (e_m)^r)$  of 1-arcs and edges a *rotation*.

**Lemma 7.** *Assume that  $(S, S_1, \dots, S_l)$  is a stable half-partnership in graph  $G$  and no 1st, 2nd and 3rd priority step is possible. If  $(e_1, (e_1)^r, e_2, (e_2)^r, \dots, e_m, (e_m)^r)$  is a rotation and  $e_i = xv \in S(v)$  then  $e_{i-1} = uw \in S^+(u)$  follows, where addition is meant modulo  $m$ . In particular, if  $e_i \in S$  then  $\{e_1, e_2, \dots, e_m\} \subseteq S$ .*

*Proof.* First we show that  $e_{i-1}^r = uv \notin S^-(v) \cup D_v(S^-(v))$ . Indirectly, assume  $e_{i-1}^r = uv \in S^-(v) \cup D_v(S^-(v))$ . If  $f = zv \in A^-(v)$  is a 1-arc then  $f$  (being a first choice) cannot be dominated at  $z$ , so  $f \in S^-(v) \cup D_v(S^-(v))$  follows, that is,

$$A^-(v) \subseteq S^-(v) \cup D_v(S^-(v)) . \tag{1}$$

By Lemma 4,

$$\begin{aligned} e_i &= (e_{i-1})_r^r \in D_v(A^-(v) \cup \{(e_{i-1})^r\}) \\ &\subseteq D_v(S^-(v) \cup D_v(S^-(v)) \cup \{(e_{i-1})^r\}) = D_v(S^-(v)) \end{aligned}$$

so  $e_i \notin S(v)$ , a contradiction. Thus  $e_{i-1}^r = uv \notin S^-(v)$ , hence  $e_{i-1}^r = uv \notin S^+(u)$  and  $e_{i-1}^r \notin D_v(S^-(v))$ , hence  $e_{i-1}^r \in D_u(S^+(u))$ . As  $e_{i-1}^r$  is the  $E(u)$ -replacement of first choice  $e_{i-1}$ , it follows that  $e_{i-1} \in S^+(u)$ , as Lemma 7 claims. □

**Lemma 8.** *Assume that no 1st, 2nd and 3rd priority step is possible for the actual graph and that  $(S, S_1, \dots, S_l)$  is a stable half-partnership. If  $(e_1, (e_1)^r, e_2, (e_2)^r, \dots, e_m, (e_m)^r)$  is a rotation then sets  $\{e_1, e_2, \dots, e_m\}$  and  $\{e_1^r, e_2^r, \dots, e_m^r\}$  are either disjoint or identical.*

*If  $\{e_1, e_2, \dots, e_m\} = \{e_1^r, e_2^r, \dots, e_m^r\}$  then  $m$  is odd and  $\{e_1, e_2, \dots, e_m\}$  is one of the  $S_j$ 's.*

*If sets  $\{e_1, e_2, \dots, e_m\}$  and  $\{e_1^r, e_2^r, \dots, e_m^r\}$  are disjoint and  $e_i \in S$  then  $\{e_1, \dots, e_m\} \subseteq S \setminus (S_1 \cup \dots \cup S_l)$ . Moreover,  $(S', S_1, \dots, S_l)$  is a stable half-partnership for  $S' = S \setminus \{e_1, e_2, \dots, e_m\} \cup \{e_1^r, e_2^r, \dots, e_m^r\}$ .*

We omit the proof due to lack of space.

**4th priority (rotation elimination) step:** Find a rotation  $(e_1, (e_1)^r, e_2, (e_2)^r, \dots, e_m, (e_m)^r)$  with disjoint sets  $\{e_1, e_2, \dots, e_m\}$  and  $\{e_1^r, e_2^r, \dots, e_m^r\}$  and delete  $\{e_1, e_2, \dots, e_m\}$ .

To justify the rotation elimination step, we only have to check that it does not create a new stable half-partnership.

**Lemma 9.** *Any stable half-partnership after a 4th priority step is also a stable half-partnership before this step.*

*Proof.* Observe that after the elimination, each edge  $(e_i)^r$  becomes a 1-arc. Assume that  $(S, S_1, \dots, S_l)$  is a stable half-partnership after the step. As in the proof of Lemma 8, let  $R^-$  denote the set of deleted 1-arcs of our rotation that enter a fixed vertex  $v$ , let  $T^- := \{(e_{i-1})^r : (e_{i-1})_r^r = e_i \in R^-\}$  be the new 1-arcs entering  $v$  and let  $A^-$  be the set of those 1-arcs that enter  $v$  and have not been deleted during the step.

By the definition of the rotation, from property (II) and the monotonicity of  $D_v$  we get  $R^- \subseteq D_v(A^- \cup R^- \cup T^-) = D_v(A^- \cup T^-) \subseteq D_v(S^-(v) \cup D_v(S^-(v))) = D_v(S^-(v))$ , and this is exactly what we wanted to prove. □

The following theorem finishes the proof of Theorem 4.

**Theorem 5.** *If no 1st, 2nd, 3rd and 4th priority step can be made on graph  $G$  then  $(E(G), S_1, S_2, \dots, S_l)$  is a stable half-partnership, where cycles  $S_i$  are given by the rotations.*

*Proof.* We have already seen that if all 1-arcs are bidirected then we have a stable partnership, which is a special case of a stable half-partnership. So assume that no further step can be executed but we still have a 1-arc  $e$  that is not bidirected. We have also seen that if we follow the alternating sequence of non bidirected 1-arcs and their replacements  $e, e^r, e_r^r, (e_r^r)^r, (e_r^r)_r^r, \dots$  then we find a rotation  $(e_1, (e_1)^r, e_2, (e_2)^r, \dots, e_m, (e_m)^r)$ , that must be an odd cycle  $S_i$  that cannot be eliminated. This means that  $m = 2k + 1$ , and  $e_i = (e_{i+k})^r = (e_{i-1})_r^r$  for



$1 \leq i \leq m$ , where addition is modulo  $m$ . We shall prove that our starting point, 1-arc  $e$  is an edge of this rotation. Hence, if our algorithm cannot make a step then all 1-arcs are either bidirected or belong to a unique odd rotation.

To this end, we may assume that  $e_r^r$  is an edge of the rotation, namely  $e_r^r = uv = e_i = (e_{i+k})^r$ . That is,  $e_i$  is the  $E(v)$ -replacement of first choice  $e_{i+k}$ , that is  $C_v(E(v) \setminus \{e_{i+k}\}) \cup D_v(E(v) \setminus \{e_{i+k}\}) = E(v) \setminus \{e_{i+k}\}$ , in other words, if  $e_i \in D_v(X)$  for some subset  $X$  of  $E(v)$  then  $e_{i+k} \in X$  must hold. But the definition of  $e_r^r$  gives that  $e_i \in D_v(A^-(v) \cup \{e_r\})$ , so  $e_{i+k} \in A^-(v) \cup \{e_r\}$ . 1-arc  $e_{i+k} \in A^+(v)$  is not bidirected, hence  $e_{i+k} \notin A^-(v)$ , thus  $e_{i+k} = e^r$  is an edge of the rotation, as well.

Similarly as above,  $e_{i+k} = vw$  is the  $E(w)$ -replacement of first choice  $e_{i-1}$  of  $w$ , hence  $C_w(E(w) \setminus \{e_{i-1}\}) \cup D_w(E(w) \setminus \{e_{i-1}\}) = E(w) \setminus \{e_{i-1}\}$ . This means that if  $e_{i+k}$  is the  $E(w)$ -replacement of edge  $e$  then  $e = e_{i-1}$  must hold. So  $e$  is an edge of our rotation, and all non bidirected 1-arcs of  $G$  belong to odd rotations.

Next we prove that all edges of  $G$  are 1-arcs. If, indirectly,  $e = uv$  is not a 1-arc, then  $e \in D_u(A^+(u))$  by the 1st priority step and  $e \notin D_u(A^-(u))$  by the 2nd priority step. So  $u$  is incident with certain nonbidirected 1-arcs such that these 1-arcs all belong to odd rotations, and each 1-arc of  $A^-(u)$  is the  $E(u)$ -replacement of different 1-arcs of  $A^+(u)$ . As  $e \notin D_u(A^-(u))$ , we have  $e \in C_u(A^-(u) \cup \{e\})$ , and  $|C_u(A^-(u) \cup \{e\})| \leq |C_u(E(u))| = |C_u(A^+(u))| = |C_u(A^-(u))|$  implies that there is a unique 1-arc  $f \in A^-(u)$  that is dominated:  $f \in D_u(A^-(u) \cup \{e\})$ . But  $f$  is the  $E(u)$ -replacement of some other arc  $g \in A^+(u)$ , and we have already seen twice in this proof that this means that  $g$  is a member of each edge set that  $C_v$ -dominates  $f$ :  $g \in A^-(u) \cup \{e\}$ . But this is a contradiction as 1-arc  $g$  is not bidirected and leaves  $u$  and  $e$  is not a 1-arc.

So if the algorithm cannot make any further step then our graph consists of bidirected 1-arcs and odd rotations  $S_1, S_2, \dots, S_k$ . It is trivial from the definition that  $(E(G), S_1, \dots, S_k)$  is a stable half-partnership.  $\square$

## 4 Complexity Issues

Irving's original algorithm [11] is very efficient: it runs in linear time. However, this algorithm is different from the one that we get if we apply our algorithm to an ordinary stable roommates problem. The difference is that Irving's algorithm has two phases: in the first phase it makes only 1st and 2nd priority steps, and after the 1st phase is over, it keeps on eliminating rotations, and never gets back to the 1st phase. The explanation is that our rotation elimination is more restricted than Irving's that does not only delete just first choices, but also removes some other edges.

Actually, it is rather straightforward to modify our algorithm to work similarly, and this improves even its time-complexity. The reason that we did not do this in the previous section is that the proof is more transparent this way. So what do we have to do to speed up the algorithm?

Observe that after a rotation elimination (4th priority) step, if 1-arc  $e_i = uv$  is deleted, then  $e_i$  ceases to be a first choice of  $u$ . The new first choice instead of



$e_i$  will be its replacement  $(e_i)^r$ . So we can include (with no extra cost) that we orient each replacement edges. Of course, refusal (2nd priority) steps may still be possible, but only at those vertices that the newly created 1-arcs enter. The definition of a rotation implies that if we apply a refusal step at such a vertex then no 1-arc gets deleted, but we might delete some unoriented arcs. So if we modify the rotation elimination step in such a way that we also include these extra 1st and 2nd priority steps within the rotation elimination step, then once we start to eliminate rotations, we never go back to the first phase. That is, we shall never have to make a proposal or a rejection step again.

To analyse the above (modified) algorithm, we have to say something about the calculation of the choice function and the dominance function. Assume that functions  $C_v$  and  $D_v$  are given by an oracle for all vertices  $v$  of  $G$ , such that for an arbitrary subset  $X$  of  $E(v)$  these oracles output  $C_v(X)$  and  $D_v(X)$  in unit time. Note that if we have only the oracle for  $D_v$  then we can easily construct one for  $C_v$  from the identity  $C_v(X) = X \setminus D_v(X)$ . Similarly, if we have an oracle for  $C_v$  then  $D_v(X)$  can be calculated by  $O(n)$  calls of the  $C_v$ -oracle, according to the definition of  $D_v$ . (As usual,  $n$  and  $m$  denotes the number of vertices and edges of  $G$ , respectively).

The algorithm starts with  $n$   $C$ -calls, and continues with  $n$   $D$ -calls. After this, each deletion in a 2nd priority step involves one  $C$ -call at vertex  $u$  where we deleted, and, if this  $C$ -call generates a new 1-arc  $e = uv$ , then we also have to make one  $D$ -call at the other end  $v$  of  $e$ . So the first phase (the 1st and 2nd priority steps) uses at most  $O(n + m)$   $C$ -calls and  $O(n + m)$   $D$ -calls.

In the second phase, the algorithm makes 3rd priority steps and modified rotation elimination steps. We do it in such a way that we start from a nonbidirected 1-arc  $e$  and follow the sequence  $e, e^r, e_r^r, (e_r^r)^r, \dots$ , until we find a rotation. The rotation will be a suffix of this sequence, and after eliminating this rotation, we reuse the prefix of this sequence, and continue the rotation search from there. This means that for the 3rd type steps we need altogether  $O(m)$   $C$ -calls. The modified rotation elimination steps consist of deleting each 1-arc  $e_i = uv$  of the rotation, orienting edges  $(e_i)^r = uw$  and applying refusal steps at vertices  $w$ . As we delete at most  $m$  edges in all rotation eliminations, this will add at most  $O(m)$   $D$ -calls. All additional work of the algorithm can be allocated to the oracle calls, so we got the following.

**Theorem 6.** *If we modify the rotation elimination step as described above, then our algorithm uses  $O(n + m)$   $C$ -calls and  $D$ -calls to find a stable half-partnership and runs in linear time.*

In the introduction, we indicated that in case of bipartite graphs there always exists a stable partnership for so called path independent substitutable choice functions (see [6]). That is, we do not have to require the increasing property of functions  $C_v$  if we want to solve the stable partnership problem on a bipartite graph. A natural question is if it is possible to generalize our result on stable partnerships to substitutable, but not necessarily increasing choice functions. In our proof, we heavily used the fact that if no proposal and rejection steps can be made then each 1-arc has a replacement and these replacements improve some

other 1-arc at their other vertices. This property is not valid in the more general setting. Below we show that the stable partnership problem for substitutable choice functions is NP-complete by reducing the 3-SAT problem to it.

**Theorem 7.** *For any 3-CNF expression  $\phi$ , we can construct a graph  $G_\phi$  and substitutable choice functions  $C_v$  on the stars of  $G_\phi$  in polynomial time in such a way that  $\phi$  is satisfiable if and only if there exists a stable partnership in  $G_\phi$  for the choice functions  $C_v$ .*

*Proof.* Define directed graph  $D_\phi$  such that  $D_\phi$  has three vertices  $a_C, b_C$  and  $v_C$  for each clause  $C$  of  $\phi$  and two vertices  $t_x$  and  $f_x$  for each variable  $x$  of  $\phi$ . The arc set of  $D_\phi$  consists of arcs  $t_x f_x$  and  $f_x t_x$  for each variable  $x$  of  $\phi$ , arcs of type  $v_C t_x$  (and  $v_C f_x$ ) if literal  $x$  (literal  $\bar{x}$ ) is present in clause  $C$  of  $\phi$ . Moreover, we have arcs  $a_C b_C, b_C v_C$  and  $v_C a_C$  for each clause  $C$  of  $\phi$ . If  $A$  is a set of arcs incident with some vertex  $v$  of  $D_\phi$  then  $C'_v(A) = A$  if no arc of  $A$  leaves  $v$ , otherwise  $C'_v(A)$  is the set of arc of  $A$  that leave  $v$ . It is easy to check that choice function  $C'_v$  is substitutable. Let  $G_\phi$  be the undirected graph that corresponds to  $D_\phi$  and let  $C_v$  denote the choice function induced by  $C'(v)$  on the undirected edges of  $G_\phi$ . We shall show that  $\phi$  is satisfiable if and only if there is a stable partnership of  $G_\phi$  for choice functions  $C_v$ , that is, if and only if there is a subset  $S$  of arcs of  $D_\phi$  such that  $S$  does not contain two consecutive arcs and for any arc  $uv$  outside  $S$  there is an arc  $vw$  of  $S$ .

Assume now that  $\phi$  is satisfiable, and consider an assignment of logical values to the variables of  $\phi$  that determine a truth evaluation of  $\phi$ . If the value of variable  $x$  is true then add arc  $f_x t_x$ , if it is false, then add arc  $t_x f_x$  to  $S$ . Do this for all variables of  $\phi$ . Furthermore, add all arcs  $a_C b_C$  to  $S$ . If variable  $x$  is true then add all arcs  $v_C t_x$  to  $S$  for all clauses that contain variable  $x$ . If variable  $y$  is false then add all arcs  $v_C f_x$  to  $S$  for all clauses that contain negated variable  $\bar{y}$ . Clearly, the hence defined  $S$  does not contain two consecutive arcs. If some arc of type  $t_x f_x$  or  $f_x t_x$  is not in  $S$  then it is dominated by the other, which is in  $S$ . Each arc of type  $v_C a_C$  is dominated by arc  $a_C b_C$  of  $S$  and each arc of type  $b_C v_C$  is dominated by some arc of type  $x t_x$  or  $y f_y$  as  $C$  has a variable that makes  $C$  true.

To finish the proof, assume that  $S$  is a stable partnership of  $D_\phi$ . Observe that for each variable  $x$  either  $t_x f_x$  or  $f_x t_x$  belongs to  $S$ , as no other arc dominates these arcs. If  $t_x f_x \in S$  then set variable  $x$  to be false, else assign logical value true to  $x$ . We have to show that for this assignment the evaluation of each clause  $C$  is true, that is, there is an arc of  $S$  from  $v_C$  to some  $t_x$  or  $f_x$ . Indirectly, if there is no such arc then the corresponding edges of  $S$  form a stable partnership on directed circuit  $v_C a_C b_C$ , which is impossible.

So the decision problem of the existence of a stable partnership is NP-complete. □

Note that though Theorem 7 shows that the stable partnership problem is difficult for non-increasing choice functions, it does not imply that Theorem 4 fails for general substitutable choice functions. Actually, the increasing property is encoded into the definition of a stable half-partnership, as replacements of a

single element cannot contain more than one element. So, in this sense Theorem 4 is not true for a directed cycle of length three if we add a parallel copy to each edge and use choice functions from the proof of Theorem 7. However, there is a natural way to extend the definition of a stable half-partnership such that the generalizability of Theorem 4 makes more sense. As we cannot state here anything nontrivial, we do not go into the details.

## 5 A Coloring Property of Stable Partnerships

We prove a generalization of a result by Cechlárová and Fleiner 3.

**Theorem 8.** *Let  $S$  be a stable partnership for graph  $G = (V, E)$  and increasing substitutable choice functions  $C_v$ . For each vertex  $v$  it is possible to partition  $E(v)$  into (possibly empty) parts  $E_0(v), E_1(v), E_2(v), \dots, E_{|S(v)|}(v)$  in such a way that for any stable partnership  $S'$  we have  $S' \cap E_0(v) = \emptyset$  and  $|S' \cap E_i(v)| = 1$  holds for  $i = 1, 2, \dots, |S(v)|$ .*

*Proof.* Let us find some stable partnership  $S$  by the algorithm in the previous section. Fix a vertex  $v$  and determine the partition of  $E(v)$  in the following manner. Each element of  $S(v)$  will belong to a different part. Follow the algorithm backwards, that is, we start from  $S$  and we build up the original  $G$  by adding edges according to the deletions of the algorithm. If we add an edge that is not incident with  $v$ , then we do not do anything. If we add an edge  $e$  of  $E(v)$  that was deleted by a 2nd priority step, then we put  $e$  into part  $E_0(v)$ . This is a good choice, since  $e$  is contained in no stable partnership. If  $e$  was deleted in a 4th priority step along a rotation then this rotation contains another (replacement) edge  $f$  incident with  $v$ . Lemma 8 shows that if we assign  $e$  to that part  $E_i(v)$  that contains  $f$ , then still no stable partnership can contain two edges of the same part  $E_i(v)$ . Let us build up the graph by backtracking the algorithm. This way, we find a part for each edge of  $E(v)$ , and this partition clearly has the property we need. □

The following corollary describes the phenomenon that is called the “rural hospital theorem” in the stable matching literature. This states that if a hospital cannot fill up its quota with residents in some stable outcome, then no matter which stable outcome is selected, it always receives the same applicants.

**Corollary 2.** *If  $S$  and  $S'$  are stable partnerships then  $|S(v)| = |S'(v)|$  for any vertex  $v$  of the underlying graph.*

There is an aesthetic problem with Theorem 8, namely, that part  $E_0(v)$  of the star of  $v$  is redundant in the following sense. If we remove all edges from  $E_0(v)$  and independently from one another we assign each of them to an arbitrary part  $E_i(v)$  (for  $1 \leq i \leq |S(v)|$ ) then the resulted partition also satisfies the requirements of Theorem 8 and  $E_0(v) = \emptyset$  for all vertices  $v$ . In what follows, by proving a strengthening of Theorem 8, we exhibit an interesting connection between the stable partnership problem and the stable roommates problem.

If  $G = (V, E)$  is a graph and  $v$  is a vertex of it then *detaching  $v$  into  $k$  parts* is the inverse operation of merging  $k$  vertices into one vertex. That is, we delete vertex  $v$ , introduce new vertices  $v^1, v^2, \dots, v^k$  and each edge that was originally incident with  $v$  will be incident with one of  $v^1, v^2, \dots, v^k$ . If  $k : V \rightarrow \{1, 2, 3, \dots\}$  is a function then a  *$k$ -detachment of  $G$*  is a graph  $G^k$  that we get by detaching each vertex  $v$  of  $G$  into  $k(v)$  parts. Clearly, there is a natural correspondence between the edges of  $G$  and that of  $G^k$ . With this notation, there is an equivalent formulation of Theorem 8: if  $G$  is a graph, and increasing substitutable choice function  $C_v$  is given for each vertex  $v$  of  $G$  and  $S$  is a stable partnership then there exists a  $k$ -detachment  $G^k$  of  $G$  in such a way that any stable partnership of  $G$  corresponds to a matching of  $G^k$ , where  $k(v) := |S(v)|$  for each vertex  $v$  of  $G$ .

**Theorem 9.** *Let  $S$  be a stable partnership for graph  $G = (V, E)$  and increasing substitutable choice functions  $C_v$ . Let  $k(v) := \max\{|S(v)|, 1\}$ . There is a  $k$ -detachment  $G^k$  of  $G$  and there are linear orders  $<_{v_i}$  on the stars of  $G^k$  such that any stable partnership of  $G$  corresponds to a stable matching of  $G^k$ .*

We omit the proof. Note that Theorem 9 is not an equivalence: it is not true that for any stable partnership problem there exists a detachment with appropriate linear orders in such a way that stable partnerships correspond bijectively to stable matchings. A counterexample is a graph on two vertices, four parallel edges with opposite linear orders on the vertices. The choice function of both vertices is the best two edges of the offered set, that is the stable partnership problem is a stable 2-matching problem. It is easy to see that there are exactly 3 different stable partnerships, but any 2-detachment has 1, 2 or 4 stable matching.

## 6 Conclusion, Open Questions

In this work, we extended Tan’s result [17] from stable matchings to a much more general framework. An interesting, and perhaps not well enough understood feature of Tan’s characterization is that it follows from Scarf’s lemma [14] just like its generalization to stable  $b$ -matchings [2]. Our present extension is a characterization of a very similar kind. A most natural question to ask is whether our result can also be deduced from an appropriate generalization of Scarf’s lemma.

Note that Scarf’s proof of his well known lemma is algorithmic. However, Scarf’s algorithm is not known to be polynomial. For special applications, like Tan’s characterization there are known polynomial algorithms, like the one of Tan and Hsueh [18]. Another natural question is whether the Tan-Hsueh algorithm can be extended to an efficient algorithm that finds a stable half-partnership. In fact, the Tan-Hsueh algorithm can be generalized to the stable  $b$ -matching problem (that is, to the many-to-many stable roommates problem) the following way. We assign quota  $b(v) = 0$  to each agent  $v$ , such that  $\emptyset$  is a

stable  $b$ -matching. Then, in each phase of the algorithm, we rise one quota by one and find a new stable half- $b$ -matching. The work we do during such a phase is essentially the same that the Tan-Hsueh algorithm does in one phase. We do this until we reach the real quota for each agent. Note that this generalized Tan-Hsueh algorithm can be extended to our stable partnership setting. We sketch it below.

For a linear order  $\prec_v$  on  $E(v)$ , define truncated choice function  $C_v^i$  on subset  $X$  of  $E(v)$  as the  $\prec_v$ -smallest  $i$  elements of  $C_v(X)$ . If  $C_v$  is increasing and substitutable then it is not difficult to choose  $\prec_v$  in such a way that  $C_v^i$  is also an increasing substitutable choice function for any  $i$ . So  $\emptyset$  is a stable half-partnership for choice functions  $C_v^0$ . In the beginning of a phase, we start with a stable half-partnership for truncated choice functions  $C_{v_1}^{i_1}, C_{v_2}^{i_2}, \dots, C_{v_n}^{i_n}$ . We pick a vertex  $v_j$  such that  $i_j < n$ , and find a stable half-partnership for the same truncated choice functions, except that we use  $C_{v_j}^{i_j+1}$  instead of  $C_{v_j}^{i_j}$ . To do this, we use a straightforward generalization of the Tan-Hsueh algorithm. After  $i_j = n$  for all vertices  $v_i$ , we have a stable half-partnership for the original problem.

## References

1. Aharoni, R., Fleiner, T.: On a lemma of Scarf. *J. Combin. Theory Ser. B* 87(1), 72–80 (2003)
2. Biró, P.: Stable  $b$ -matchings on graphs, Master's thesis, Budapest university of Technology and Economics (2003)
3. Cechlárová, K., Fleiner, T.: On a generalization of the stable roommates problem. *ACM Trans. Algorithms* 1(1), 143–156 (2005)
4. Crawford, V.P., Knoer, E.M.: Job matching with heterogeneous firms and workers. *Econometrica* 49, 437–450 (1981)
5. Feder, T.: A new fixed point approach for stable networks and stable marriages. *J. Comput. System Sci.* 45(2), 233–284 (1992); Twenty-first Symposium on the Theory of Computing (Seattle, WA, 1989)
6. Fleiner, T.: A fixed-point approach to stable matchings and some applications. *Math. Oper. Res.* 28(1), 103–126 (2003)
7. Fleiner, T.: The stable roommates problem with choice functions, EGRES Technical Report TR-2007-11 (November 2007), <http://www.cs.elte.hu/egres>
8. Fleiner, T.: On the stable  $b$ -matching polytope. *Math. Social Sci.* 46(2), 149–158 (2003)
9. Gale, D., Shapley, L.S.: College admissions and stability of marriage. *Amer. Math. Monthly* 69(1), 9–15 (1962)
10. Gusfield, D., Irving, R.W.: The stable marriage problem: structure and algorithms. MIT Press, Cambridge (1989)
11. Irving, R.W.: An efficient algorithm for the “stable roommates” problem. *J. Algorithms* 6(4), 577–595 (1985)
12. Kelso Jr., A.S., Crawford, V.P.: Job matching, coalition formation, and gross substitutes. *Econometrica* 50, 1483–1504 (1982)
13. Roth, A.E., Oliveira Sotomayor, M.A.: Two-sided matching. Cambridge University Press, Cambridge (1990)
14. Scarf, H.E.: The core of an  $N$  person game. *Econometrica* 35, 50–69 (1967)

15. Sethuraman, J., Teo, C.-P., Qian, L.: Many-to-one stable matching: geometry and fairness. *Math. Oper. Res.* 31(3), 581–596 (2006)
16. Subramanian, A.: A new approach to stable matching problems. *SIAM J. Comput.* 23(4), 671–700 (1994)
17. Tan, J.J.M.: A necessary and sufficient condition for the existence of a complete stable matching. *J. Algorithms* 12(1), 154–178 (1991)
18. Tan, J.J.M., Hsueh, Y.C.: A generalization of the stable matching problem. *Discrete Appl. Math.* 59(1), 87–102 (1995)
19. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. *Pacific J. of Math.* 5, 285–310 (1955)

# A New Approach to Splitting-Off

Attila Bernáth<sup>1,\*</sup> and Tamás Király<sup>2,\*\*</sup>

<sup>1</sup> Dept. of Operations Research, Eötvös University, Pázmány P. s. 1/C, Budapest, Hungary H-1117. The author is a member of the Egerváry Research Group (EGRES)

bernath@cs.elte.hu

<sup>2</sup> MTA-ELTE Egerváry Research Group, Department of Operations Research, Eötvös University, Pázmány Péter sétány 1/C, Budapest, Hungary, H-1117

tkiraly@cs.elte.hu

**Abstract.** A new approach to undirected splitting-off is presented in this paper. We study the behaviour of splitting-off algorithms when applied to the problem of covering a symmetric skew-supermodular set function by a graph. This hard problem is a natural generalization of many solved connectivity augmentation problems, such as local edge-connectivity augmentation of graphs, global arc-connectivity augmentation of mixed graphs with undirected edges, or the node-to-area connectivity augmentation problem in graphs. Using a simple lemma we characterize the situation when a splitting-off algorithm can be stuck. This characterization enables us to give very simple proofs for the classical results mentioned above. Finally we apply our observations in generalizations of the above problems: we consider two connectivity augmentation problems in hypergraphs. The first is the local edge-connectivity augmentation of undirected hypergraphs by hyperedges of minimum total size without increasing the rank. The second is global arc-connectivity augmentation of mixed hypergraphs by adding hyperedges of minimum total size without increasing the rank. We show that a greedy approach works in (almost) all of these cases.

## 1 Introduction

Let us be given a finite ground set  $V$ . A set function  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  is called *skew-supermodular* if at least one of the following two inequalities holds for every  $X, Y \subseteq V$ :

$$\begin{aligned} p(X) + p(Y) &\leq p(X \cap Y) + p(X \cup Y), & (\cap\cup) \\ p(X) + p(Y) &\leq p(X - Y) + p(Y - X). & (-) \end{aligned}$$

In this paper we consider the problem of covering a symmetric skew-supermodular set function  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  by a graph, or in some cases by a hypergraph of restricted type. We distinguish two versions of this problem. In the **degree specified version** we are also given a **degree specification**  $m : V \rightarrow \mathbb{Z}_+$  and the question is whether a graph (or hypergraph)  $G$  covering  $p$

---

\* Supported by OTKA grants K60802 and TS 049788.

\*\* Supported by grants OTKA K60802 and OMF01608/2006.

exists with  $d_G(v) = m(v)$  for every  $v \in V$ . In the **minimum version** we simply want to find a graph covering  $p$  that has a minimum number of edges (for hypergraphs we want to minimize the sum of the sizes of the hyperedges). Possibly the latter problem seems more interesting and natural, however a solution for the first always gives a solution to the second by the skew-supermodularity of  $p$ , therefore we will mainly speak about the degree specified problem.

This problem is a natural generalization of many connectivity augmentation problems. Examples include the *local edge-connectivity augmentation problem in graphs* solved by Frank [5], the *global edge-connectivity augmentation problem in mixed graphs* solved by Bang-Jensen, Frank and Jackson [1], and the **node-to-area connectivity augmentation problem** solved by Ishii and Hagiwara [7]. The general problem of covering a symmetric skew-supermodular set function  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  with a minimum number of graph edges is known to be NP-complete (see e.g. [4], where the NP-completeness of the degree specified version is also shown implicitly). However, as seen above, many special cases have been shown to be polynomially solvable. The key approach in solving this kind of questions is the technique called “splitting-off”: we first find a smallest number of graph edges covering  $p$  that are incident to a new node  $s$ , and then we try to get rid of  $s$  by splitting off pairs of edges incident to it (which means that we substitute this path of length 2 with the shortcut). We have found a new approach to this second step that simplifies proofs for known results and enables us to prove new results, too. The key lemma (Lemma 2) of our results states that if there is a set  $X \subseteq V$  with  $p(X) \geq 2$  then there always exists an **admissible splitting** (the splitting is admissible if the resulting graph still covers  $p$ ). Consider a greedy algorithm that starts with a graph containing edges incident to  $s$  and in each step performs an (arbitrary) admissible splitting as long as it is possible. Then Lemma 2 enables us to prove some interesting properties of the situation when this algorithm gets stuck.

We analyze these properties in Section 3. We consider two special symmetric skew-supermodular functions in Subsections 3.3 and 3.4. The first case is when  $p(X) = \max\{q(X), q(\overline{X})\}$  with a crossing supermodular function  $q$  (which includes the global edge-connectivity augmentation of a mixed graph or hypergraph). Here we obtain a very good characterization of the stuck situation. It turns out that if we contract tight sets then  $q(X) = 1$  or  $q(\overline{X}) = 1$  for any nonempty  $X \subsetneq V$ . Introducing the notation  $\mathcal{F} = \{X \subseteq V : q(X) = 1\}$  and  $\text{co}(\mathcal{F}) = \{X \subseteq V : \overline{X} \in \mathcal{F}\}$  this leads to the following question: how does a crossing family  $\mathcal{F} \subseteq 2^V - \{\emptyset, V\}$  look like that satisfies  $\mathcal{F} \cup \text{co}(\mathcal{F}) = 2^V - \{\emptyset, V\}$ ? We give a complete characterization of such families in Theorem 9. This theorem enables us to show a result on global arc-connectivity augmentation of mixed hypergraphs in Section 4.3, but we find it interesting for its own sake, too.

A set function  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  is called *crossing negamodular* if  $(-)$  holds whenever  $X$  and  $Y$  are crossing. The second special symmetric skew-supermodular function is defined by  $p(X) = \max\{q(X), q(\overline{X})\}$  with a crossing negamodular function  $q$  (which is a generalization of the function arising in the node-to-area connectivity augmentation problem). Covering such a function with



a minimum number of graph edges already includes NP-complete problems, as was observed by Miwa and Ito [9]. So we make a similar assumption to that of Ishii and Hagiwara and assume that  $q = R - d_G$  where  $R$  is a crossing negamodular function that does not take 1 as value and  $G$  is an arbitrary graph. We show that for such a function  $q$  the greedy approach will always produce a graph that has at most one more edge than the optimum.

In Section 4 we give applications. First, we demonstrate the strength of our approach by giving simple proofs for the classical splitting theorem of Mader [8] (used by Frank in [5]) and the undirected splitting theorem in mixed graphs used by Bang-Jensen, Frank and Jackson in [1].

In Section 4.2 we prove the following result (Theorem 14): the *local edge-connectivity augmentation problem of hypergraphs with hyperedges of minimum total size* can be solved by adding only graph edges and one hyperedge whose size is at most the rank of the original hypergraph. There is only one exceptional case when this is impossible: when the minimum total size is odd and we augment a graph. This theorem can be regarded as a common generalization of the theorem of Frank [5] on local edge-connectivity augmentation of graphs and the theorem of Szigeti [10] on local edge-connectivity augmentation of hypergraphs.

Finally in Section 4.3 we consider **global arc-connectivity augmentation of a mixed hypergraph without increasing the rank** by undirected hyperedges. We show that the greedy approach can fail for this problem, but only slightly. To be more precise, we prove that a mixed hypergraph of rank at most  $\gamma$  can always be augmented greedily to become  $(k, l)$ -arc-connected from a specified root node  $r$  if  $k, l \geq 2$  by graph edges and a hyperedge of size at most  $\gamma + 1$ .

## 2 Preliminaries

Let us be given a finite ground set  $V$ . For subsets  $X, Y$  of  $V$  let  $\overline{X}$  be  $V - X$  (the ground set will be clear from the context). If  $X$  has only one element  $x$  then we will call it a **singleton** and we will not distinguish between  $X$  and its only element  $x$ . Sets  $X, Y \subseteq V$  are **intersecting** if  $X \cap Y, X - Y$  and  $Y - X$  are all nonempty. If furthermore  $X \cup Y \neq V$  then we say that they are **crossing**. For a family  $\mathcal{F} \subseteq 2^V$  let  $\text{co}(\mathcal{F}) = \{X \subseteq V : \overline{X} \in \mathcal{F}\}$ . We say that  $\mathcal{F}$  is a **ring family (crossing family)** if  $X, Y \in \mathcal{F}$  implies  $X \cap Y, X \cup Y \in \mathcal{F}$  for an arbitrary (crossing, resp.) pair  $X, Y$ .

Let  $q : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  be a set function: we will require all the set functions in this paper to satisfy  $q(\emptyset) \leq 0$  and  $q(V) \leq 0$ . Define **the complement of  $q$**  as  $\overline{q}(X) = q(\overline{X})$  and **the symmetrized of  $q$**  by  $q^s(X) = \max\{q(X), q(\overline{X})\}$  for any  $X \subseteq V$ . Observe that the symmetrized of a skew-supermodular function is skew-supermodular.

A set function is *symmetric* if  $p(X) = p(V - X)$  for every  $X \subseteq V$ . Any function  $m : V \rightarrow \mathbb{R}$  also induces a set function (that will also be denoted by  $m$ ) with the definition  $m(X) = \sum_{v \in X} m(v)$  for any  $X \subseteq V$ .

For a hypergraph  $H = (V, \mathcal{E})$  and a set  $X \subseteq V$  we define  $d_H(X) = |\{e \in \mathcal{E} : e \text{ enters } X\}|$  (the **degree** of  $X$  in  $H$ ). This is a symmetric submodular function. For two set functions  $d, p$  we say that  $d$  **covers**  $p$  if  $d(X) \geq p(X)$  for any  $X \subseteq V$

( $d \geq p$  for short). We say that the hypergraph  $H$  **covers**  $p$  if  $d_H$  covers  $p$ . Observe that  $H$  covers  $p$  if and only if  $H$  covers  $p^s$ . The **total size of the hypergraph** is the sum of the cardinalities of the hyperedges: if our hypergraph is a graph then this is two times the number of the edges of this graph. The **rank of a hypergraph** is the size of the largest hyperedge in it. For  $S, T \subseteq V$  let  $\lambda_H(S, T)$  denote the maximum number of edge-disjoint paths starting at a vertex of  $S$  and ending at a vertex of  $T$  (we say that  $\lambda_H(S, T) = \infty$  if  $S \cap T \neq \emptyset$ ). By Menger's theorem

$$\lambda_H(S, T) = \min\{d_H(X) : T \subseteq X \subseteq V - S\}.$$

A **mixed graph** may have directed and undirected edges, too. For a mixed graph  $G$  and sets  $X, Y \subseteq V$  let  $d_G(X, Y)$  denote the number of (undirected or directed) edges of  $G$  with one endpoint in  $X - Y$  and the other in  $Y - X$ .

For a set function  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  we introduce the polyhedron

$$C(p) = \{x \in \mathbb{R}^V : x(Z) \geq p(Z) \forall Z \subseteq V, x \geq 0\}.$$

It is known that for a skew-supermodular function  $p$  this is an (integer) contrapolymatroid (for details see [1]). We assume in the whole article that we can test membership in polynomial time in  $C(p - d_G)$  for any graph  $G$ : this will be sufficient to turn our results into polynomial algorithms and this will always hold in the applications given below.

In what follows let  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  a symmetric, skew-supermodular function that satisfies  $p(\emptyset) \leq 0$  and  $m : V \rightarrow \mathbb{Z}$  a nonnegative function satisfying  $m(X) \geq p(X)$  for any  $X \subseteq V$  (i.e. an integer element of  $C(p)$ ). We would like to decide whether there is a graph (or possibly hypergraph)  $G$  covering  $p$  that satisfies  $d_G(v) = m(v)$  for every  $v \in V$ . We note that, by the properties of a contrapolymatroid, a polynomial algorithm to the degree specified covering problem will give rise to a solution to the minimum version of the problem, and to more general versions such as the minimum node-cost problem. For more details we refer to [1]. We say that  $m \in C(p) \cap \mathbb{Z}^V$  is **minimal** if  $m' \in C(p) \cap \mathbb{Z}^V$ ,  $m' \leq m$  implies that  $m' = m$ .

For a node  $v \in V$  we say that  $v$  is **positive** if  $m(v) > 0$ , and **neutral** otherwise. The set of positive nodes will be denoted by  $V^+$ . Assume  $u, v \in V^+$  are two positive nodes (possibly  $u = v$ , but then  $m(u) \geq 2$  is assumed). The operation **splitting-off (at  $u$  and  $v$ )** is the following: let

$$m' = m - \chi_{\{u\}} - \chi_{\{v\}} \text{ and } p' = p - d_{(V, \{uv\})}. \tag{1}$$

One can observe that this is indeed the usual notion of splitting-off: if we introduce a graph  $G = (V + s, E)$  with every edge of  $E$  incident to  $s$  and  $d_G(s, v) = m(v)$  for any  $v \in V$  then we are back at the well known splitting-off operation. However we found this way of presenting our results more convenient. If  $m'(X) \geq p'(X)$  for any  $X \subseteq V$  then we say that the splitting off is **admissible**. Clearly, splitting off at  $u$  and  $v$  is admissible if and only if there is no dangerous set  $X$  containing both  $u$  and  $v$  (a set  $X$  is **dangerous** if  $m(X) - p(X) \leq 1$  and it is called **tight** if  $m(X) - p(X) = 0$ ). We will also say that such a dangerous

set  $X$  **blocks the splitting at  $u$  and  $v$** , or simply that  $X$  **blocks  $u$  and  $v$** . The presentation is simpler if we don't allow creating loops, i.e. we only split off at distinct nodes. This is always possible except for some trivial cases (e.g. when  $m$  is positive on only one node). In particular, if  $m$  is minimal, then an admissible splitting can not create a loop.

We consider the following class of algorithms to find a graph  $G$  covering  $p$  with degree function  $m$ . The algorithm does successive admissible splitting steps (with possibly taking care of other things, too, but we assume that it only stops when no admissible splitting is possible), until it **terminates** with  $m'(V) = 0$  or **gets stuck** with  $m'(V) > 0$ . Obviously it can not get stuck with  $m'(V) = 2$  and if  $m(V)$  was odd then it cannot find a degree-specified graph, though we don't want to exclude this case since we sometimes allow hyperedges, too, instead of graph edges.

Let  $M_p = \max\{p(X) : X \subseteq V\}$ . A set  $X$  with  $p(X) = M_p$  will be called  $p$ -**maximal**. Clearly, if  $M_p \leq 0$  then any splitting-off is admissible. Note that for two  $p$ -maximal sets  $X$  and  $Y$  either both of  $X \cap Y$  and  $X \cup Y$  or both of  $X - Y$  and  $Y - X$  are also  $p$ -maximal.

If  $T \subseteq V$  then contracting  $T$  roughly means that from now on we consider it to be a singleton. Formally this means that we define  $V/T = V - T + v_T$  where  $v_T$  was not in  $V$ . For any set function  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  we define  $p/T : 2^{V/T} \rightarrow \mathbb{Z} \cup \{-\infty\}$  by  $p/T(X) = p(X)$  if  $v_T \notin X$  and  $p/T(X) = p(X - v_T + T)$  if  $v_T \in X$ . In this contracted problem a splitting-off is admissible if it is admissible with respect to  $p/T$ . Note that  $p/T$  will inherit the interesting properties of  $p$  investigated in this paper (e.g. symmetry, crossing supermodularity, skew-supermodularity etc.). A useful observation is the following.

**Lemma 1.** *Let  $u, v \in V$  with  $m(u), m(v) > 0$ . If we contract a tight set  $T$  then the splitting at  $u'$  and  $v'$  is admissible if and only if the splitting at  $u$  and  $v$  is admissible (where  $u'$  ( $v'$ ) is the contracted image of  $u$  ( $v$ , respectively)).*

*Proof.* By the definition of  $p/T$  if the splitting-off at  $u$  and  $v$  was admissible then it clearly stays admissible. Let us prove the other direction. Assume that  $u', v'$  becomes admissible while  $u, v$  was not admissible, i.e. there was a set  $X \subseteq V$  with  $p(X) \geq m(X) - 1$  with  $u, v \in X$ . Clearly, neither  $T \subseteq X$  nor  $X \cap T = \emptyset$  can hold. If  $(\cap \cup)$  holds for  $X$  and  $T$  then  $X \cup T$  is also dangerous, a contradiction. So  $(-)$  must hold for them, meaning  $X - T$  is also dangerous and  $u, v \in X - T$ , a contradiction again. □

This allows us to simplify some of the proofs by assuming that every tight set is a singleton.

### 3 Characterization of the Stuck Case

The starting point of our results is the following lemma.

**Lemma 2.** *Let  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  be a symmetric, skew-supermodular function and  $m \in C(p) \cap \mathbb{Z}^V$ . If  $M_p > 1$  then there is an admissible splitting.*

*Proof.* Let  $Y$  be a minimal set satisfying  $p(Y) = M_p$ . By symmetry,  $p(V - Y) = M_p$ , too, so we can choose a minimal set  $Z \subseteq V - Y$  satisfying  $p(Z) = M_p$ . Since  $M_p \geq 1$  we can choose  $y \in Y, z \in Z$  with  $m(y), m(z) > 0$ . We claim that the splitting at  $y$  and  $z$  is admissible. Assume not and consider a dangerous set  $X$  containing  $y$  and  $z$ . Since  $m(X - Y) < m(X)$ ,  $X$  and  $Y$  cannot satisfy  $(-)$ , since this would imply  $p(Y - X) = M_p$ , contradicting the minimality of  $Y$ . So  $X$  and  $Y$  must satisfy  $(\cap\cup)$ , which implies (using  $m(X \cap Y) < m(X)$ ) that  $p(X \cup Y) = M_p$  and  $m(X - Y) = 1$ . Now  $X \cup Y$  and  $Z$  cannot satisfy  $(-)$  since this would give  $p(Z - (X \cup Y)) = M_p$ , contradicting the minimality of  $Z$ . Therefore  $X \cup Y$  and  $Z$  satisfy  $(\cap\cup)$  implying that  $p(Z \cap (X \cup Y)) = M_p$ , which is only possible if  $Z \subseteq X \cup Y$ . But  $2 \leq M_p = p(Z) \leq m(Z) \leq m(X - Y) = 1$  gives a contradiction.  $\square$

Let us mention some important consequences of this lemma. Firstly, if there is no admissible splitting-off, then  $p \leq 1$  and every pair  $u, v \in V^+$  is in a dangerous set  $X$ : this means that  $p(X) = 1$  and  $m(X) = 2$ , hence  $m \leq 1$ . Furthermore, we are only forced to create a loop in an admissible splitting-off if  $|V^+| = 1$  (so by the symmetry of  $p$  this implies  $M_p \leq 0$ ): we assume in the rest of the paper that this is not the situation.

**Corollary 3.** *If  $p$  is a symmetric, skew-supermodular function and  $m \in C(p) \cap \mathbb{Z}^V$  then there is a hypergraph  $H$  covering  $p$  with degree function  $m$  that contains at most one hyperedge of size at least 3.*

Consider the following greedy algorithm.

Algorithm GREEDYCOVER

begin

**INPUT** A symmetric skew-supermodular function  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  (given with an oracle) and  $m \in C(p) \cap \mathbb{Z}^V$ .

**OUTPUT** A graph  $G = (V, E)$  and a hyperedge  $e$  such that the hypergraph  $G + e$  covers  $p$  and  $d_{G+e}(v) = m(v)$  for every  $v \in V$ .

1.1. Initialize  $G = (V, \emptyset)$ .

1.2. While there exists an admissible pair  $u, v$  do

1.3. Let  $m = m - \chi(u) - \chi(v)$  and  $p = p - d_{(V, \{(u,v)\})}$  and  $G = G + (uv)$ .

1.4. Output  $G$  and  $e$  where  $\chi_e = m$ .

end

Clearly, if one can test membership in  $C(p - d_G)$  for any graph  $G$  in polynomial time then this algorithm terminates in polynomial time. In the following sections we will consider this algorithm for special symmetric skew-supermodular set functions and we will say something about the size of the hyperedge in its output. We say that the algorithm **got stuck** if the hyperedge in the output is of size greater than 0.

### 3.1 Greedy Problems

The preceding observations led us to the following definition. Consider either the minimum or the degree specified version of covering a symmetric skew-supermodular function  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  with some hypergraph of restricted

type (e.g. a graph). Define the **greedy bound** by  $gb(p) = \max\{\sum_{i=1}^t p(X_i) : \mathcal{X}$  is a subpartition of  $V\} = \min\{1 \cdot x : x \in C(p)\}$ : this is obviously a lower bound for the minimum total size of any hypergraph covering  $p$ . We call such a covering problem **greedily solvable for  $m$**  if the algorithm GREEDYCOVER necessarily finds the desired hypergraph (where  $m \in C(p) \cap \mathbb{Z}^V$  is assumed). We call it **greedily solvable** if GREEDYCOVER outputs a suitable hypergraph for any  $m \in C(p) \cap \mathbb{Z}^V$  (if we only allow graphs then we make the additional assumption that  $m(V)$  is even).

Let us give an example. It has been proved by Benczúr and Frank [2] that the problem of covering a symmetric, crossing supermodular set function by a minimum number of graph edges can be solved in polynomial time. In a special case this problem can be solved greedily: the proof will be given later. We note that this theorem includes the global edge-connectivity augmentation problem of graphs solved by Watanabe and Nakamura [11].

**Theorem 4.** *If  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  is symmetric, crossing supermodular, does not take 1 as value, and  $G = (V, E)$  is an arbitrary graph, then the problem of covering  $p' = p - d_G$  by a graph is greedily solvable.*

Many proofs below consider the situation when the greedy algorithm gets stuck. In most of the cases we can assume that this is already the case in the beginning, since after some steps we are again at an instance of our starting problem: an example of this is Theorem 4.

### 3.2 General Observations on the Stuck Case

We can read out many things about the situation when the algorithm GREEDYCOVER gets stuck from Lemma 2. Assume that the procedure started with the function  $p_0$  and  $m_0 \in C(p_0) \cap \mathbb{Z}^V$ , performed some admissible splittings and got stuck at some point: let the graph of the edges split so far be  $G$  and let  $p = p_0 - d_G$  and  $m(v) = m_0(v) - d_G(v)$  for any  $v \in V$ . If there is no admissible splitting, then every pair  $u, v \in V^+$  is in a dangerous set  $X$ : since  $p \leq 1$  this means that  $p(X) = 1$  and  $m(X) = 2$ , hence  $m \leq 1$ . In the rest of this section we assume that we are at this stuck situation. The interesting case for us will be the case when the splitting procedure gets stuck with  $m(V) \geq 4$ : we will assume this in the rest of this section. In some cases it is also useful to assume that tight sets are singletons: we will always state this explicitly.

We can notice that with  $m(V) - 1$  edges we can cover the function  $p$ : any spanning tree on  $V^+$  will cover  $p$ . We could possibly cover  $p$  with less edges, however we can give a lower bound: one needs at least  $\lceil 2(m(V) - 1)/3 \rceil$  edges to finish the procedure. For a proof see [3].

Let us give a lemma that will be useful later. Assume  $x_0, x_1, x_2 \in V$  are three different positive nodes and  $X_0, X_1, X_2$  are the three dangerous sets blocking them with  $x_i \in X_j \cap X_k$  for any  $\{i, j, k\} = \{0, 1, 2\}$ . (Since we assume that  $m(V) \geq 4$  the three sets  $X_0, X_1, X_2$  are pairwise crossing here.) We will say that  $X_0, X_1$  and  $X_2$  form a **blocking-triangle**.  $X_2$  will be called **slim** if  $X_0 \cap X_1 \cap X_2 = \emptyset$  and  $X_2 - (X_0 \cup X_1) = \emptyset$ .

**Lemma 5 (Slimming Lemma).** *Assume that  $X_0$  and  $X_1$  satisfy  $(\cap\cup)$ . Then  $(X_2 - (X_0 \cap X_1)) \cap (X_0 \cup X_1)$  is also dangerous and blocks  $x_0, x_1$ .*

*Proof.* Since  $X_0$  and  $X_1$  satisfy  $(\cap\cup)$ ,  $p(X_0 \cap X_1) = p(X_0 \cup X_1) = 1$ . Now  $X_0 \cap X_1$  and  $X_2$  cannot satisfy  $(\cap\cup)$ , since that would imply that  $p(X_0 \cap X_1 \cap X_2) = 1$ , but  $m(X_0 \cap X_1 \cap X_2) = 0$ . This implies that  $p(X'_2) = 1$  where  $X'_2 = X_2 - (X_0 \cap X_1)$ . Now  $X'_2$  and  $X_0 \cup X_1$  cannot satisfy  $(-)$ , since that would give  $p(X'_2 - (X_0 \cup X_1)) = 1$  contradicting  $m(X'_2 - (X_0 \cup X_1)) = 0$ . So we obtain from  $(\cap\cup)$  that  $p(X'_2 \cap (X_0 \cup X_1)) = 1$  and clearly  $x_0, x_1 \in X'_2 \cap (X_0 \cup X_1)$ .  $\square$

For the subsequent two subsections let us introduce some notations. If  $p$  is the symmetrized of a function  $q$  then for any set  $X$  either  $p(X) = q(X)$  or  $p(X) = q(\bar{X})$  (possibly both). In the former case we say that  $X$  is of  $q$ -**type**, in the latter we say that  $X$  is of  $\bar{q}$ -**type** (so  $X$  can be of both types). Assume that for any pair  $x, y$  of positive nodes we fix a dangerous set  $X(x, y)$  blocking them. We introduce two (undirected, simple) graphs on the set of positive nodes: the edge set of the  $q$ -**graph** ( $\bar{q}$ -**graph**) consists of the pairs  $u, v$  of positive nodes having  $q(X(u, v)) = 1$  ( $\bar{q}(X(u, v)) = 1$ , respectively). Since there is no admissible splitting, the union of these two graphs is the complete graph (on the set of positive nodes), and an edge may belong to both graphs. We will call this 2-edge-coloured complete graph **the  $q\bar{q}$ -graph**.

### 3.3 Crossing Supermodular Functions

In this subsection we assume that  $p$  is the symmetrized of a crossing supermodular function  $q$ . A set function  $q : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  is called *crossing supermodular* if it satisfies  $(\cap\cup)$  whenever  $X$  and  $Y$  are crossing: note that a symmetric crossing supermodular function is also skew-supermodular (which is not necessarily the case without the symmetry). Furthermore, a symmetric crossing supermodular function satisfies both  $(\cap\cup)$  and  $(-)$  if  $X$  and  $Y$  are crossing. One can check that the complement of a crossing supermodular function is also crossing supermodular, and the symmetrized of a crossing supermodular function is skew-supermodular.

If two crossing sets  $X$  and  $Y$  are of the same type then they will satisfy  $(\cap\cup)$ . If furthermore  $p(X) = p(Y) = 1$  then their intersection and union is also of the same type as  $X$  and  $Y$  (here we use that  $p \leq 1$ ). On the other hand if  $X$  and  $Y$  are of different types then  $p(X - Y) = p(Y - X) = 1$ . Also note that from any three sets there are two of the same type.

In this subsection we will also assume that tight sets are singletons. If  $p$  is symmetric and crossing supermodular, then it is easy to check that every node is positive (one can find examples showing that this does not hold in general, if only the skew-supermodularity of  $p$  is assumed). However we will prove this in a more general case, namely when  $p$  is the symmetrized of a crossing supermodular function  $q$ . First it is useful to prove the following lemma.

**Lemma 6.** *If  $p$  is the symmetrized of a crossing supermodular function  $q$  then any set blocking two positive nodes contains only these two nodes.*

*Proof.* Assume that there is a set  $X$  with  $2 = m(X) = p(X) + 1$  that also contains neutral nodes. By possibly complementing  $q$  we can assume that  $X$  is of  $q$ -type. Let  $x, z \in X$  be the positive nodes and let  $y \in V - X$  be another positive node. Let  $Y$  be a set blocking  $x$  and  $y$ . We claim that  $Y$  must be of  $q$ -type, too. If not, then  $X - Y = z, Y - X = y$ , since they are tight. But then consider any set  $Z$  blocking  $y$  and  $z$ .  $Z$  cannot be of  $q$ -type (since this would imply  $Z \cap X = z$  and  $Y - Z = y$ , a contradiction), neither of  $\bar{q}$ -type (for a similar reason). So we proved that for any positive  $y \in V - X$  the set  $Y(y)$  blocking  $x$  and  $y$  is of  $q$ -type. So the union of these sets  $Y' = \cup_{y \in V - X, m(y) > 0} Y(y)$  is also of  $q$ -type, and has  $p(Y') = q(Y') = 1$ . However this implies that  $1 = p(V - Y') = m(z) = m(V - Y')$ , so it is tight, but this is a contradiction together with  $|X| > 2$  (note that  $Y' \cap X = x$ ).  $\square$

The lemma implies that the edge set of the  $q$ -graph ( $\bar{q}$ -graph) consists of the pairs  $u, v$  of positive nodes having  $q(\{u, v\}) = 1$  ( $\bar{q}(\{u, v\}) = 1$ , respectively). Observe that a non-singleton connected component  $X \neq V$  of the  $q$ -graph is also of  $q$ -type and has  $q(X) = 1$  (and similarly for the  $\bar{q}$ -graph). This immediately implies the result promised before.

**Lemma 7.** *If  $p$  is the symmetrized of a crossing supermodular function  $q$  then every node is positive.*

*Proof.* Suppose not, then the set of positive nodes  $V^+ \neq V$  must be connected in at least one of the two graphs (since the union of two disconnected graphs cannot be the complete graph), so  $p(V^+) = 1$ . But then  $p(V - V^+) = 1$  by the symmetry, contradicting  $m(V - V^+) = 0$ .  $\square$

What is more, this implies the following surprising observation.

**Lemma 8.** *If  $p$  is the symmetrized of a supermodular function  $q$ , then  $p(X) = 1$  for any  $X$  with  $\emptyset \neq X \neq V$  (i.e.  $q(X) = 1$  or  $q(V - X) = 1$  for every such set).*

*Proof.* By the preceding argument, any non-singleton  $X \subsetneq V$  must be connected in at least one of the two graphs, so has  $p(X) = 1$  (it is also easy to see for singletons, using  $m(V) \geq 4$ ).  $\square$

Consequently we have a crossing family  $\mathcal{F}$  containing all sets with  $q$  value 1, and the family of the complements of this family  $\text{co}(\mathcal{F})$  (these are the sets with  $\bar{q}$  value 1), and the union of these two families is  $2^V - \{\emptyset, V\}$ . In the following theorem we will characterize such families (for sake of brevity we will also add  $\emptyset$  and  $V$  in the family: we can always add to or remove from a crossing family these sets). It turns out that the graphs introduced above contain almost all information about the family in question.

Let  $x \in V$  and let  $X_1, \dots, X_t$  be  $t \geq 1$  pairwise disjoint subsets of  $V - x$  (possibly  $t = 1$  and  $X_1 = \emptyset$ ). We introduce the following family:

$$\mathcal{F}_{x, X_1, \dots, X_t} = \{X \subseteq V : x \in X \text{ or } X \subseteq X_i \text{ for some } i \in 1, \dots, t\}.$$



**Theorem 9.** *Let  $\mathcal{F} \subseteq 2^V$  be a crossing family with  $\emptyset, V \in \mathcal{F}$  that satisfies  $\mathcal{F} \cup \text{co}(\mathcal{F}) = 2^V$ . Then either  $V$  has exactly four elements and  $\mathcal{F} = 2^V \setminus \{\{y, z\}\}$  for some  $y \neq z, y, z \in V$  or there exists a node  $x \in V$  and  $X_1, \dots, X_t$  pairwise disjoint subsets of  $V - x$  for some  $t \geq 1$  such that either  $\mathcal{F}$  or  $\text{co}(\mathcal{F})$  is equal to  $\mathcal{F}_{x, X_1, \dots, X_t}$  or  $\mathcal{F}_{x, X_1, \dots, X_t} \cup \{V - x\}$ .*

Due to space limitations, we omit the proof of this theorem, it can be found in [3]. A simple corollary that is worth mentioning is the following.

**Theorem 10.** *Let  $\mathcal{F} \subseteq 2^V$  be a ring family with  $\emptyset, V \in \mathcal{F}$  that satisfies  $\mathcal{F} \cup \text{co}(\mathcal{F}) = 2^V$ . Then there exists a node  $x \in V$  and a (possibly empty) set  $X_1 \subseteq V - x$  such that either  $\mathcal{F}$  or  $\text{co}(\mathcal{F})$  is equal to  $\mathcal{F}_{x, X_1}$ .*

We finish this subsection by proving Theorem [4].

*Proof (Proof of Theorem [4]).* Assume that the greedy algorithm gets stuck (at start). Let  $p' = p - d_G$ . We can assume that tight sets are singletons and that  $m(V) \geq 4$ .  $G$  cannot be empty, since  $p$  does not take 1 as value. Let  $(x, y)$  be an arbitrary edge of  $G$  and  $z \in V - x - y$ . Consider the crossing sets  $X = \{x, z\}$  and  $Y = \{y, z\}$ . By  $(\cap \cup)$ , we have the contradiction:

$$2 = p'(X) + p'(Y) \leq p'(X \cap Y) + p'(X \cup Y) - 2 \leq 0. \quad \square$$

### 3.4 Crossing Negamodular Functions

A set function  $p : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  is called *crossing negamodular* if  $(-)$  holds whenever  $X$  and  $Y$  are crossing. Note that the symmetrized of a crossing negamodular function is skew-supermodular, but the complement of a crossing negamodular function is not crossing negamodular. An important special case is a **monotone decreasing** function: by that we mean a function  $p$  that satisfies  $p(\emptyset) \leq 0$  but  $p(X) \geq p(Y)$  for any  $\emptyset \subsetneq X \subseteq Y \subseteq V$ .

An important observation is the following: if  $q : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  is crossing negamodular and  $X, Y \subseteq V$  are crossing sets with  $q(X) = \bar{q}(Y) = M_q$ , then  $q(X \cap Y) = M_q$  and  $\bar{q}(X \cup Y) = M_q$ .

**Theorem 11.** *Let  $R : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$  be a crossing negamodular function that does not take 1 as value, and  $G = (V, E)$  an arbitrary graph. Let  $q = R - d_G$  and  $p = q^s$ . Then  $p$  can be covered by  $\lceil gb(p)/2 \rceil + 1$  edges greedily.*

*Proof.* We will prove more: we show that there always exists an admissible splitting-off if  $m(V) \geq 5$ . Let  $m \in C(p) \cap \mathbb{Z}^V$  and assume that the greedy algorithm is stuck (already at start) and tight sets are singletons. We will prove that  $m(V) \leq 4$ , so assume indirectly that  $m(V) > 4$ . We claim that there is a triangle in either the  $q$ -graph or the  $\bar{q}$ -graph (where we fix a dangerous set  $X(u, v)$  blocking  $u$  and  $v$  for any  $u, v \in V^+$ ): this is clear if  $m(V) \geq 6$  by a theorem of Ramsey. If  $m(V) = 5$  then one can check that the only way to avoid this is when the  $q$ -graph and the  $\bar{q}$ -graph form two edge-disjoint cycles on 5 nodes. However, if we assume that the  $q$ -graph is the cycle on  $v_1, v_2, v_3, v_4, v_5$  in



this order, then the union of  $X(v_1, v_2)$  and  $X(v_1, v_4)$  is of  $\bar{q}$ -type, so resetting  $X(v_3, v_5) := V - (X(v_1, v_2) \cup X(v_1, v_4))$  will give a 3-cycle on  $v_3, v_4$  and  $v_5$  in the  $q$ -graph.

Consider such a triangle  $x_0, x_1, x_2$  of positive nodes (say a “blue triangle”, where blue is either the colour of the  $q$ -graph or that of the  $\bar{q}$ -graph): by the assumptions made so far there is a set  $X \subseteq V - \{x_0, x_1, x_2\}$  such that the set  $X(x_i, x_j)$  blocking  $x_i$  and  $x_j$  is  $X + x_i + x_j$  for any distinct  $i, j \in \{0, 1, 2\}$ . Let  $Y = X + \{x_0, x_1, x_2\}$ . By the properties of  $q$  we have  $p(x_i) = 1$  for any  $i$ , and of course  $p(Y - x_i) = 1$  for any  $i$ . This means that  $d_G(x_i) > 0$  and  $d_G(Y - x_i) > 0$  for any  $i$ , since  $R$  does not take 1 as value. But  $d_G(Y)$  must be 0, otherwise we could not have  $(-)$  with equality for every  $x_i, x_j$ . Consider any positive node  $u$  distinct from  $x_0, x_1, x_2$ . Suppose that the edge  $(u, x_0)$  is blue again. This implies that  $X(u, x_0) = X + u + x_0$  and using that  $(-)$  must hold with equality for  $X(u, x_0)$  and  $X(x_0, x_1)$  gives that  $d_G(x_2, x_1) = d_G(x_2) > 0$ . This immediately implies that at most one blue edge can go from  $u$  to the set  $\{x_0, x_1, x_2\}$  in the  $q\bar{q}$ -graph. So taking another positive node  $v$  distinct from  $u, x_0, x_1, x_2$  we see that there is an  $i$  such that the edges  $u, x_i$  and  $v, x_i$  are both red in the  $q\bar{q}$ -graph. Again this means that there exists a set  $Z \subseteq V - Y - \{u, v\}$  such that  $X(u, x_i) = Z + u + x_i$  and  $X(v, x_i) = Z + v + x_i$ , but then using that  $(-)$  must hold with equality for these two sets we obtain that  $d_G(x_i) = d_G(x_i, Y - x_i) = 0$ , a contradiction.  $\square$

As an application of this theorem consider the node-to-area connectivity augmentation solved by Ishii and Hagiwara [7]. The problem is the following. Given a graph  $G = (V, E)$ , a collection of subsets  $\mathcal{W}$  of  $V$  and a function  $r : \mathcal{W} \rightarrow \mathbb{Z}_+$ , find a minimum number of new edges  $F$  such that  $\lambda_{G+F}(x, W) \geq r(W)$  for any  $W \in \mathcal{W}$  and  $x \in V$ . This problem is in general NP-complete, so the authors assume that  $r \geq 2$  and give a polynomial time algorithm that solves this problem. Let us show why this problem is a special case of the problem addressed in Theorem 1. Define  $R(X) = \max\{r(W) : W \in \mathcal{W}, W \cap X = \emptyset\}$  for any  $\emptyset \neq X \subseteq V$  and  $R(\emptyset) = 0$ . Then this is a monotone decreasing function, so it is crossing negamodular, and it does not take 1 as value, hence it belongs to the class of functions considered in Theorem 1. We mention that  $R^s$  is the function that was called a **symmetric semi-monotone function** in [6]. Our results do not characterize the cases when the greedy bound for covering  $R - d_G$  can be achieved, but they imply that a greedy algorithm can only fail by at most one for this problem.

## 4 Applications

### 4.1 Simple Proofs

**Theorem 12 (Mader’s lemma).** *Let  $G = (V + s, E)$  be a graph such that there is no cut edge incident to  $s$  and  $d_G(s) > 3$ . Then there exists a splitting-off at  $s$  that preserves the local edge-connectivities in  $V$ .*

*Proof.* If there is no cut edge incident to  $s$  then  $\lambda_G(u, v) \geq 2$  for any pair of  $s$ -neighbours  $u, v$ . Let us define  $R(X) = \max\{\lambda_G(x, y) : x \in X, y \in V - X\}$  for any  $X$  with  $\emptyset \neq X \neq V$ ,  $R(\emptyset) = R(V) = 0$ , and  $p(X) = R(X) - d_{G[V]}(X)$  for any  $X \subseteq V$ . Let  $m(v) = d_G(s, v)$  for any  $v \in V$ . It is well known and easy to check that  $(R$  and  $)p$  is a symmetric and skew-supermodular function. By assumption,  $m$  covers  $p$ . Assume that there is no splitting-off and consider a blocking triangle  $X, Y, Z$ . We can also assume that tight sets are singletons. Consider the following two cases.

**Case I:**  $X$  and  $Y$  satisfy  $(\cap \cup)$ . By Lemma 5 we can assume that  $Z$  is slim. Since there is no cut edge incident to  $s$ , both  $R(X \cap Z)$  and  $R(Y \cap Z)$  are at least two, implying that their degree in  $G$  is at least two. Let  $R(Z) = \lambda_G(z, v)$  with  $z \in Z$  and  $v \in V - Z$  and assume wlog. that  $z \in X \cap Z$ . Then  $d_G(Z) - 1 \leq R(Z) \leq R(X \cap Z) \leq d_G(X \cap Z) = d_G(Z) - d_G(Y \cap Z) + d_G(X \cap Z, Y \cap Z) \leq d_G(Z) - 2 + d_G(X \cap Z, Y \cap Z)$  implies that  $d_G(X \cap Z, Y \cap Z) > 0$ , but then  $X$  and  $Y$  cannot satisfy  $(\cap \cup)$  with equality.

**Case II:**  $X, Y$  and  $Z$  pairwise satisfy  $(-)$ . Since tight sets are singletons,  $|W - X| = |W - Y| = |W - Z| = 1$  where  $W = X \cup Y \cup Z$ . Using that there is a neighbour of  $s$  not in  $W$  we can deduce that  $R(W) \geq 2$ . However, since a pair of these three sets must satisfy  $(-)$  with equality, there must not be an edge of  $G[V]$  leaving  $W$ . But this would imply that  $p(W) \geq 2$ , contradicting Lemma 2. □

Next we will give a simple proof of a theorem of Bang-Jensen, Frank and Jackson 11 on undirected splitting-off in mixed graphs: the  $k = l$  case is a special case of Theorem 3.2 of 11, so we also manage to extend slightly this special case.

**Theorem 13 (Bang-Jensen, Frank, Jackson).** *Let  $M = (V + s, E)$  be a mixed graph and assume that  $s$  is only incident with undirected edges. Let  $r \in V$  and  $k, l \geq 2$  integers and assume that  $\lambda_M(r, v) \geq k$  and  $\lambda_M(v, r) \geq l$  for any  $v \in V$ . Then there exists a splitting-off at  $s$  preserving this property, provided that  $d_M(s) > 3$ .*

*Proof.* We can assume that  $M - s$  is a digraph (by substituting undirected edges by oppositely directed pairs of arcs): let us denote this digraph by  $D = (V, A)$  and let  $m(v) = d_M(s, v)$  for any  $v \in V$ . Let the function  $q$  be defined by  $q(\emptyset) = q(V) = 0$ ,  $q(X) = k - \varrho_D(X)$  for any nonempty  $X \subseteq V - r$  and  $q(X) = l - \varrho_D(X)$  for any  $X \subsetneq V$  with  $r \in X$ . Then one can check that  $q$  is crossing supermodular; let  $p = q^s$ . Since  $M$  is  $(k, l)$ -arc-connected from  $r$  (apart from  $s$ ),  $m(X) \geq p(X)$  for any  $X \subseteq V$ . Assume that there is no splitting-off and tight sets are singletons. Then by Lemma 6 every vertex is positive. Consider a blocking triangle  $X, Y, Z$  with  $r \notin X \cup Y \cup Z$ . We can assume without loss of generality that  $X$  and  $Y$  are both of the same type, implying that  $d_D(X, Y) = 0$ . By Lemma 5 we can assume that  $Z$  is slim. Then either  $\varrho_D(Z) = k - 1$  or  $\delta_D(Z) = l - 1$ : assume the former, the other case being analogous. But  $\varrho_D(Z \cap X) \geq k - 1$  and  $\varrho_D(Z \cap Y) \geq k - 1$  together with  $k \geq 2$  implies that  $d_D(X, Y) > 0$ , a contradiction. □

### 4.2 Local Edge-Connectivity Augmentation of Hypergraphs

In this section we consider the *local edge-connectivity augmentation of hypergraphs without increasing the rank*. Let  $H = (V, \mathcal{E})$  be a hypergraph of rank at most  $\gamma$ , and let  $r : V \times V \rightarrow \mathbb{Z}_+ \setminus \{1\}$  be a symmetric edge-connectivity requirement that does not take 1 as value. Let us define the set function  $R(X)$  as  $R(\emptyset) = R(V) = 0$  and

$$R(X) = \max_{u \in X, v \notin X} r(u, v) \quad (\emptyset \neq X \subsetneq V).$$

Our aim is to find a hypergraph  $H'$  of minimum total size such that  $H + H'$  covers  $R$ , that is,  $\lambda_{H+H'}(u, v) \geq r(u, v)$  for every pair of nodes  $u, v$ . Since  $R$  is a skew supermodular function, the algorithm GREEDYCOVER defined at the beginning of Section 3 gives a solution that contains graph edges and at most one hyperedge. The question we want to answer is whether the size of this hyperedge is at most  $\gamma$ . One case when this is obviously not true is when  $\gamma = 2$  and the greedy bound is odd: then the size of the hyperedge will be 3. The following theorem shows that this is the only exceptional case. Note that this theorem generalizes the theorem of Frank [5] on local edge-connectivity augmentation of graphs.

**Theorem 14.** *Let  $H = (V, \mathcal{E})$  be a hypergraph of rank at most  $\gamma > 2$ , and let  $r : V \times V \rightarrow \mathbb{Z}_+ \setminus \{1\}$  be a symmetric edge-connectivity requirement. Then the greedy algorithm gives a solution to the minimum total size local edge-connectivity augmentation problem that contains only graph edges and one hyperedge of size at most  $\gamma$ .*

Due to space limitations, we omit the proof of this theorem, it can be found in [3].

### 4.3 Global Arc-Connectivity Augmentation of Mixed Hypergraphs

A **mixed hypergraph**  $M = (V, \mathcal{A})$  is a pair of a finite set  $V$  and a family  $\mathcal{A}$  of subsets of  $V$  (repetitions are allowed). For an  $a \in \mathcal{A}$  every  $v \in a$  can be either a **head node**, a **tail node** or even both (**head-tail node**), such that every hyperarc contains at least one head and one tail. More formally we could say that  $\mathcal{A}$  contains nonempty ordered set-pairs  $(T, H)$  ( $T$  being the set of tails,  $H$  being the heads, possibly  $H \cap T \neq \emptyset$ ). An undirected hypergraph can be considered (for our purposes) as a special mixed hypergraph where every node in a hyperarc is a head-tail node of this hyperarc. The set  $V$  is called the **node set** of the mixed hypergraph, the family  $\mathcal{A}$  is called the **hyperarc set** (or sometimes shortly the **arc set**) of the mixed hypergraph. **Reversing a hyperarc in  $\mathcal{A}$**  means switching the roles of the nodes in it, i.e. head nodes become tail nodes and vice versa (so head-tail nodes remain like that). When we say that  $v$  is a tail node of a hyperarc  $a$  then we also allow that it is a head-tail node (and similarly for head nodes).

In a mixed hypergraph  $M$ , a **path** between nodes  $s$  and  $t$  is an alternating sequence of distinct nodes and hyperarcs  $s = v_0, a_1, v_1, a_2, \dots, a_k, v_k = t$ , such

that  $v_{i-1}$  is a tail node of  $a_i$  and  $v_i$  is a head node of  $a_i$  for all  $i$  between 1 and  $k$ . A hyperarc  $a$  **enters a set**  $X$  if there is a head node of  $a$  in  $X$  and there is a tail node of  $a$  in  $V - X$ . A hyperarc **leaves a set** if it enters the complement of this set. For a set  $X$  we define  $\varrho_M(X) = |\{a \in \mathcal{A} : a \text{ enters } X\}|$  (the **in-degree** of  $X$ ) and  $\delta_M(X) = \varrho_M(V - X)$  (the **out-degree** of  $X$ ). It is easy to check that the functions  $\varrho$  and  $\delta$  are submodular functions. Given a mixed hypergraph  $M = (V, \mathcal{A})$  and sets  $S, T \subseteq V$ , let  $\lambda_M(S, T)$  denote the maximum number of arc-disjoint paths starting at a vertex of  $S$  and ending at a vertex of  $T$  (we say that  $\lambda_M(S, T) = \infty$  if  $S \cap T \neq \emptyset$ ). By Menger’s theorem:

$$\lambda_M(S, T) = \min\{\varrho_M(X) : T \subseteq X \subseteq V - S\}.$$

If  $M = (V, \mathcal{A})$  is a mixed hypergraph,  $r \in V$  is a designated root node and  $k, l$  are nonnegative integers, then we say that  $M$  is  **$(k, l)$ -arc-connected from  $r$**  if  $\lambda_M(r, v) \geq k$  and  $\lambda_M(v, r) \geq l$  for any  $v \in V$ . Let us define the set function  $q = q_{M,r,k,l}$  by  $q(\emptyset) = q(V) = 0$ ,  $q(X) = k - \varrho_M(X)$  for any nonempty  $X \subseteq V - r$  and  $q(X) = l - \varrho_M(X)$  for any  $X \subsetneq V$  with  $r \in X$ . Then one can check that  $q$  is crossing supermodular. For a hypergraph  $H$  one can prove that  $M + H$  is  $(k, l)$ -arc-connected from  $r$  if and only if  $d_H$  covers  $q$  (or equivalently  $q^s$ ).

Let  $M = (V, \mathcal{A})$  be a mixed hypergraph and let  $k, l \geq 2$  be integers. We assume that  $M$  is of rank at most  $\gamma$ . We want to make  $M$   $(k, l)$ -arc-connected by adding an undirected, degree specified hypergraph of minimum total size that also has rank at most  $\gamma$ . Is it true that the greedy bound can be achieved? Can this be done greedily? The answer is “almost yes”: an example shows that sometimes this can only be done by adding a hyperedge of cardinality  $\gamma + 1$  (even for  $k = l = 2$ ). Consider the following mixed hypergraph  $M = (V, \mathcal{A})$ : let  $|V| \geq 3$  and  $x, y \in V$  be two nodes. There are 3 hyperarcs in  $\mathcal{A}$ : one is a digraph arc  $(x, y)$ , the second is  $(y, V - x - y)$  and the third is  $(V - x - y, x)$ . Finally let  $k = l = 2$  and  $\gamma = |V| - 1$ . It is easy to see that the greedy bound is  $|V|$  and the only way to achieve it is to add the hyperedge  $V$ . However, we can prove the following result.

**Theorem 15.** *If  $M$  is of rank at most  $\gamma \geq 2$  and  $k, l \geq 2$  are integers, then we can make  $M$   $(k, l)$ -arc-connected greedily by the addition of graph edges and one hyperedge of size at most  $\gamma + 1$ .*

*Proof.* We can assume that the greedy algorithm is stuck already at start. Let  $q = q_{M,r,k,l}$  and  $p = q^s$  and let  $m \in C(p) \cap \mathbb{Z}^V$  that satisfies the greedy bound. We have to prove that  $m(V)$  is at most  $\gamma + 1$ . We can also assume that tight sets are singletons (and delete singleton hyperedges, since they are irrelevant for connectivity), so by the observations in Section 3.3 every node is positive. By Theorem 9, there is an  $x \in V$  such that (by possibly reversing every hyperarc of  $M$  and switching the role of  $k$  and  $l$ ) every set  $X \neq V$  with  $x \in X$  has  $q(X) = 1$  (observe that this consequence is also true for the sporadic example on 4 nodes). First we claim that  $V - x$  cannot contain hyperarcs. Assume that it does contain a hyperarc  $a$ , let  $v$  be an arbitrary head node of  $a$ , and let  $X = a - v + x$  and  $Y = \{v, x\}$ . These sets are crossing (since  $|a| < |V - x|$  by the assumption)

and of  $q$ -type, but  $(\cap \cup)$  cannot hold with equality for them, a contradiction. So every hyperarc of  $M$  contains  $x$ . We claim that if  $v \neq x$  is a tail of a hyperarc  $a$  satisfying  $|a| \geq 3$ , then  $x \in H_a$  and  $T_a - v - x = \emptyset$ . To see this consider the sets  $X = a - v$  and  $Y = \{v, x\}$ . Then  $q(X) = q(Y) = 1$  but one can check that  $(\cap \cup)$  cannot hold with equality for  $X$  and  $Y$ , a contradiction. So the hyperedges leaving any  $v \in V - x$  all enter  $x$ . If  $x = r$  then  $l - 1 = \varrho(x) = \sum_{v \in V - x} \delta(v) = |V - x|(l - 1)$  contradicting that  $|V| > 2$  and  $l > 1$ . On the other hand, if  $x \neq r$  then  $k - 1 = \varrho(x) = \sum_{v \in V - x} \delta(v) = (|V| - 2)(l - 1) + (k - 1)$ , again contradicting that  $|V| > 2$  and  $l > 1$ .  $\square$

## References

1. Bang-Jensen, J., Frank, A., Jackson, B.: Preserving and increasing local edge-connectivity in mixed graphs. *SIAM J. Discrete Math.* 8(2), 155–178 (1995)
2. Benczúr, A.A., Frank, A.: Covering symmetric supermodular functions by graphs. *Math. Program. Ser. B* 84(3), 483–503 (1999); Connectivity augmentation of networks: structures and algorithms (Budapest, 1994)
3. Bernáth, A., Király, T.: A new approach to splitting-off, Tech. Report TR-2008-02, Egerváry Research Group, Budapest (2008), <http://www.cs.elte.hu/egres>
4. Cosh, B., Jackson, B., Király, Z.: Local connectivity augmentation in hypergraphs is NP-complete (manuscript)
5. Frank, A.: Augmenting graphs to meet edge-connectivity requirements. *SIAM J. Discrete Math.* 5(1), 25–53 (1992)
6. Grappe, R., Szigeti, Z.: Note: Covering symmetric semi-monotone functions. *Discrete Appl. Math.* 156(1), 138–144 (2008)
7. Ishii, T., Hagiwara, M.: Minimum augmentation of local edge-connectivity between vertices and vertex subsets in undirected graphs. *Discrete Appl. Math.* 154(16), 2307–2329 (2006)
8. Mader, W.: A reduction method for edge-connectivity in graphs. *Ann. Discrete Math.* 3, 145–164 (1978); (Cambridge Combinatorial Conf., Trinity College, Cambridge, 1977)
9. Miwa, H., Ito, H.: NA-edge-connectivity augmentation problems by adding edges. *J. Oper. Res. Soc. Japan* 47(4), 224–243 (2004)
10. Szigeti, Z.: Hypergraph connectivity augmentation. *Math. Program. Ser. B* 84(3), 519–527 (1999); Connectivity augmentation of networks: structures and algorithms (Budapest, 1994)
11. Watanabe, T., Nakamura, A.: Edge-connectivity augmentation problems. *J. Comput. System Sci.* 35(1), 96–144 (1987)

# Can Pure Cutting Plane Algorithms Work?

Arrigo Zanette<sup>1</sup>, Matteo Fischetti<sup>1,\*</sup>, and Egon Balas<sup>2,\*\*</sup>

<sup>1</sup> DEI, University of Padova

zanettea@math.unipd.it, matteo.fischetti@unipd.it

<sup>2</sup> Carnegie Mellon University, Pittsburgh, PA

eb17@andrew.cmu.edu

**Abstract.** We discuss an implementation of the lexicographic version of Gomory’s fractional cutting plane method and of two heuristics mimicking the latter. In computational testing on a battery of MIPLIB problems we compare the performance of these variants with that of the standard Gomory algorithm, both in the single-cut and in the multi-cut (rounds of cuts) version, and show that they provide a radical improvement over the standard procedure. In particular, we report the exact solution of ILP instances from MIPLIB such as `stein15`, `stein27`, and `bm23`, for which the standard Gomory cutting plane algorithm is not able to close more than a tiny fraction of the integrality gap. We also offer an explanation for this surprising phenomenon.

**Keywords:** Cutting Plane Methods, Gomory Cuts, Degeneracy in Linear Programming, Lexicographic Dual Simplex, Computational Analysis.

## 1 Introduction

Modern branch-and-cut methods for (mixed or pure) Integer Linear Programs are heavily based on general-purpose cutting planes such as Gomory cuts, that are used to reduce the number of branching nodes needed to reach optimality. On the other hand, pure cutting plane methods based on Gomory cuts alone are typically not used in practice, due to their poor convergence properties.

In a sense, branching can be viewed as just a “symptomatic cure” to the well-known drawbacks of Gomory cuts—saturation, bad numerical behavior, etc. From the cutting plane point of view, however, the cure is even worse than the disease, in that it hides the real source of the troubles. So, a “theoretically convergent” method such as the Gomory one becomes ancillary to enumeration, and no attempt is made to try to push it to its limits. In this respect, it is instructive to observe that a main piece of information about the performance of Gomory cuts (namely, that they perform much better if generated in rounds)

---

\* The work of the first two authors was supported by the Future and Emerging Technologies unit of the EC (IST priority), under contract no. FP6-021235-2 (project “ARRIVAL”) and by MiUR, Italy (PRIN 2006 project “Models and algorithms for robust network optimization”).

\*\* The work of the third author was supported by National Science Foundation grant #DMI-0352885 and Office of Naval Research contract #N00014-03-1-0133.

was discovered only in 1996 (Balas, Ceria, Cornuéjols, and Natraj [2]), i.e., about 40 years after their introduction [7].

The purpose of our project, whose scope extends well beyond the present paper, is to try to come up with a viable pure cutting plane method (i.e., one that is not knocked out by numerical difficulties), even if on most problems it will not be competitive with the branch-and-bound based methods.

As a first step, we chose to test our ideas on Gomory’s fractional cuts, for two reasons: they are the simplest to generate, and they have the property that when expressed in the structural variables, all their coefficients are integer (which makes it easier to work with them and to assess how nice or weird they are). In particular, we addressed the following questions:

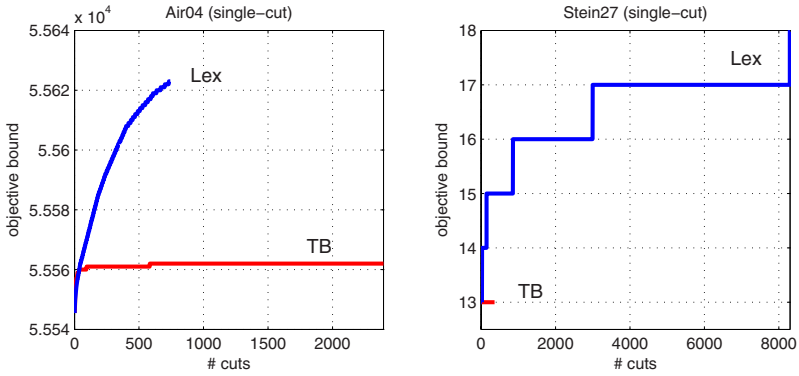
- i) Given an ILP, which is the most effective way to generate fractional Gomory cuts from the optimal LP tableaux so as to push the LP bound as close as possible to the optimal integer value?
- ii) What is the role of degeneracy in Gomory’s method?
- iii) How can we try to counteract the numerical instability associated with the iterated use of Gomory cuts?
- iv) Is the classical polyhedral paradigm “the stronger the cut, the better” still applicable in the context of Gomory cuts read from the tableau? The question is not at all naive, as one has to take into account the negative effects that a stronger yet denser (or numerically less accurate) cut has in the next tableaux, and hence in the next cuts.

As we were in the process of testing various ways of keeping the basis determinant and/or condition number within reasonable limits, our youngest coauthor had the idea of implementing the lexicographic dual simplex algorithm used in one of Gomory’s two finite convergence proofs. Gomory himself never advocated the practical use of this method; on the contrary, he stressed that its sole purpose was to simplify one of the two proofs, and that in practice other choice criteria in the pivoting sequence were likely to work better. Actually, we have no information on anybody ever having tried extensively this method in practice.

The lexicographic method has two basic ingredients: (a) the starting tableau is not just optimal, i.e., dual feasible, but lexicographically dual-feasible, and the method of reoptimization after adding a cut is the lexicographic dual simplex method; and (b) at least after every  $k$  iterations for some fixed  $k$ , the row with the first fractional basic variable is chosen as source row for the next cut.

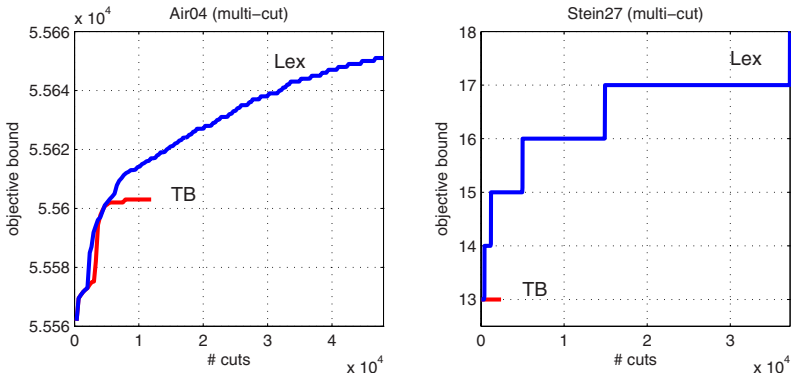
The implementation of this method produced a huge surprise: the lexicographic method produces a dramatic improvement not only in gap closure (see Figure [1]), but also in determinant and cut coefficient size.

It is well known that cutting plane methods work better if the cuts are generated in rounds rather than individually (i.e., if cuts from all fractional variables are added before reoptimization, rather than reoptimizing after every cut). Now it seems that if we are generating rounds of cuts rather than individual cuts, the use of the lexicographic rule would make much less sense, in particular because (b) is automatically satisfied—so the lexicographic rule plays a role only



**Fig. 1.** Comparison between the textbook and lexicographic implementations of single-cut Gomory’s algorithm on `air04` and `stein27`

in shaping the pivoting sequence in the reoptimization process. So we did not expect it to make much of a difference. Here came our second great surprize: as illustrated in Figure 2, even more strikingly than when using single cuts, comparing the standard and lexicographic methods with rounds of cuts shows a huge difference not only in terms of gap closed (which for the lexicographic version is 100% for more than half the instances in our testbed), but also of determinant size and coefficient size.



**Fig. 2.** Comparison between the textbook and lexicographic implementations of multi-cut Gomory’s algorithm on `air04` and `stein27`

In this paper we discuss and evaluate computationally and implementation of the lexicographic version of Gomory’s fractional cutting plane method and of two heuristics mimicking the latter one, and offer an interpretation of the outcome of our experiments.



## 2 Gomory Cuts

In this paper we focus on pure cutting plane methods applied to solving Integer Linear Programs (ILPs) of the the form:

$$\begin{aligned} \min \quad & c^T x \\ & Ax = b \\ & x \geq 0 \text{ integer} \end{aligned}$$

where  $A \in \mathbb{Z}^{m \times n}$ ,  $b \in \mathbb{Z}^m$  and  $c \in \mathbb{Z}^n$ . Let  $P := \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$  denote the LP relaxation polyhedron, that we assume be bounded.

The structure of a pure cutting plane algorithm for the solution of the ILP problem can be roughly outlined as follows:

1. solve the LP relaxation  $\min\{c^T x : x \in P\}$  and denote by  $x^*$  an optimal vertex
2. if  $x^*$  is integer, then we are done
3. otherwise, search for a *violated cut*, i.e., a hyperplane  $\alpha^T x \leq \alpha_0$  separating  $x^*$  from the convex hull of integer feasible points, add it to the original formulation, and repeat.

The cut generation is of course a crucial step in any cutting plane method, as one is interested in easily-computable yet effective cuts.

In 1958, Gomory [7] (see also [9]) gave a simple and elegant way to generate violated cuts, showing that  $x^*$  can always be separated by means of a cut easily derived from a row of the LP-relaxation optimal tableau. The cut derivation is based on a rounding argument: given any equation  $\sum_{j=1}^n \gamma_j x_j = \gamma_0$  valid for the  $P$ , if  $x$  is constrained to be nonnegative and integer then  $\sum_{j=1}^n \lfloor \gamma_j \rfloor x_j \leq \lfloor \gamma_0 \rfloor$  is a valid cut. According to Gomory's proposal, the equation is the one associated with a row of the LP optimal tableau whose basic variable is fractional: we will refer to this row as the *cut generating* row, and to the corresponding basic variable as the *cut generating* variable.

The resulting cut, called *Fractional Gomory Cut* (FGC) or Chvátal-Gomory cut, has important theoretical and practical properties. First of all, one can use FGCs read from the LP tableau to derive a finitely-convergent cutting plane method. Secondly, because of the integrality of all the cut coefficients, the associated slack variable can be assumed to be integer, so the addition of FGCs does not introduce continuous variables that could make the rounding argument inapplicable in the next iterations. Moreover, the fact that the cut coefficients are integer ensures a certain "confidence level" about the numerical accuracy of the generated cuts. Indeed, once a cut is generated, small fractionalities in the computed coefficients can be removed safely so as to reduce error propagation, whereas FGCs with significant fractionalities are likely to be invalid (because of numerical issues) and hence can be skipped.

In 1960, Gomory [8] introduced the *Gomory Mixed Integer* (GMI) cuts to deal with the mixed-integer case. In case of pure ILPs, GMI cuts are applicable as

well. Actually, GMI cuts turn out to *dominate* FGCs in that each variable  $x_j$  receives a coefficient increased by a fractional quantity  $\theta_j \in [0, 1)$  with respect to the FGCs (writing the GMI in its  $\leq$  form, with the same right-hand-side value as in its FGC counterpart). E.g, a FGC cut of the type  $2x_1 - x_2 + 3x_3 \leq 5$  may correspond to the GMI  $2.27272727x_1 - x_2 + 3.18181818x_3 \leq 5$ . So, from a strict polyhedral point of view, there is no apparent reason to insist on FGCs when a stronger replacement is readily available at no extra computational effort. However, as shown in the example above, the coefficient integrality of GMI cuts is no longer guaranteed, hence the nice numerical properties of FGCs are lost. Even more importantly, as discussed in the sequel, the introduction of “weird fractionalities” in the cut coefficients may have uncontrollable effects on the fractionality of the next LP solution and hence of the associated LP tableau. As a result, it is unclear whether FGC or GMI cuts are better suited for a pure cutting plane method based on tableau cuts—a topic that we are going to investigate in the near future.

It is important to stress that the requirement of reading (essentially for free) the cuts directly from the optimal LP tableau makes the Gomory method intrinsically different from a method that works solely with the original polyhedron where the cut separation is decoupled from the LP reoptimization, as in the recent work of Fischetti and Lodi [6] on FGCs or Balas and Saxena [5] on GMI (split) cuts. Actually, only the first round of cuts generated by the Gomory method (those read from the very first optimal tableau) work on the original polyhedron, subsequent rounds are generated from a polyhedron truncated by previously generated cuts.

We face here a very fundamental issue in the design of pure cutting plane methods based of (mixed-integer or fractional) Gomory cuts read from the LP optimal tableau. Since we expect to generate a long sequence of cuts that eventually lead to an optimal integer solution, we have to take into account side effects of the cuts that are typically underestimated when just a few cuts are used (within an enumeration scheme) to improve the LP bound. In particular, one should try to maintain a “clean” optimal tableau so as to favor the generation of “clean” cuts in the next iterations. To this end, it is important to avoid as much as possible generating (and hence cutting) LP optimal vertices with a “weird fractionality”—the main source of numerical inaccuracy. This is because the corresponding optimal LP basis necessarily has a large determinant (needed to describe the fractionality), hence the tableau contains weird entries that lead to weaker and weaker Gomory cuts.

In this respect, dual degeneracy (that is notoriously massive in cutting plane methods) can play an important role and actually can *favor* the practical convergence of the method, provided that it is exploited to choose the *cleanest* LP solution (and tableau) among the equivalent optimal ones—the optimized sequence of pivots performed by a generic LP solver during the tableau reoptimization leads invariably to an uncontrolled growth of the basis determinant, and the method gets out of control after few iterations.

### 3 Degeneracy and the Lexicographic Dual Simplex

As already mentioned, massive dual degeneracy occurs almost invariably when solving ILPs by means of cutting plane algorithms. Indeed, cutting planes tend to introduce a huge number of cuts that are almost parallel to the objective function, whose main goal is to prove or to disprove the existence of an integer point with a certain value of the objective function.

In his proof of convergence, Gomory used the lexicographic dual simplex to cope with degeneracy. The lexicographic dual simplex is a generalized version of the simplex algorithm where, instead of considering the minimization of the objective function, viewed without loss of generality as an additional integer variable  $x_0 = c^T x$ , one is interested in the minimization of the entire solution vector  $(x_0, x_1, \dots, x_n)$ , where  $(x_0, x_1, \dots, x_n) >_{LEX} (y_0, y_1, \dots, y_n)$  means that there exists an index  $k$  such that  $x_i = y_i$  for all  $i = 1, \dots, k - 1$  and  $x_k > y_k$ . In the lexicographic, as opposed to the usual, dual simplex method the ratio test does not only involve two scalars (reduced cost and pivot candidate) but a column and a scalar. So, its implementation is straightforward, at least in theory. In practice, however, there are a number of major concerns that limit this approach:

1. the ratio test has a worst-case quadratic time complexity in the size of the problem matrix;
2. the ratio test may fail in selecting the right column to preserve lex-optimality, due to round-off errors;
3. the algorithm requires taking control of each single pivot operation, which excludes the possibility of applying much more effective pivot-selection criteria.

The last point is maybe the most important. As a clever approach should not interfere too much with the black-box LP solver used, one could think of using a perturbed linear objective function  $x_0 + \epsilon_1 x_1 + \epsilon_2 x_2 \dots$ , where  $x_0$  is the actual objective and  $1 \gg \epsilon_1 \gg \epsilon_2 \gg \dots$ . Though this approach is numerically unacceptable, one can mimic it by using the following method which resembles the iterative procedure used in the construction of the so-called Balinsky–Tucker tableau [3] and is akin to the slack fixing used in sequential solution of preemptive linear goal programming (see [1] and [10]).

Starting from the optimal solution  $(x_0^*, x_1^*, \dots, x_n^*)$ , we want to find another basic solution for which  $x_0 = x_0^*$  but  $x_1 < x_1^*$  (if any), by exploiting dual degeneracy. So, we fix the variables that are nonbasic (at their bound) and have a nonzero reduced cost. This fixing implies the fixing of the objective function value to  $x_0^*$ , but has a major advantage: since we fix only variables at their bounds, the fixed variables will remain out of the basis in all the subsequent steps. Then we reoptimize the LP by using  $x_1$  as the objective function (to be minimized), fix other nonbasic variables, and repeat. The method then keeps optimizing subsequent variables, in lexicographic order, over smaller and smaller dual-degenerate subspaces, until either no degeneracy remains, or all variables

are fixed. At this point we can unfix all the fixed variables and restore the original objective function, the lex-optimal basis being associated with the non-fixed variables.

This approach proved to be quite effective (and stable) in practice: even for large problems, where the classical algorithm is painfully slow or even fails, our alternative method requires short computing time to convert the optimal basis into a lexicographically-minimal one. We have to admit however that our current implementation is not perfect, as it requires deciding whether a reduced cost is zero or not: in some (rare) cases, numerical errors lead to a wrong decision that does not yield a lexicographically dual-feasible final tableau. We are confident however that a tighter integration with the underlying LP solver could solve most of the difficulties in our present implementation.

## 4 Heuristics Variants

While the lexicographic simplex method gives an exact solution to the problem of degeneracy, simple heuristics can be devised that mimic the behavior of lexicographic dual simplex. The scope of these heuristics is to try to highlight the crucial properties that allow the lexicographic method to produce stable Gomory cuts.

As already mentioned, a lex-optimal solution can in principle be reached by using an appropriate perturbation of the objective function, namely  $x_0 + \epsilon_1 x_1 + \dots + \epsilon_n x_n$  with  $1 \gg \epsilon_1 \gg \dots \gg \epsilon_n$ . Although this approach is actually impractical, one can use a 1-level approximation where the perturbation affects a single variable only, say  $x_i$ , leading to the new objective function  $\min x_0 + \epsilon x_i$ . The perturbation term is intended to favor the choice of an equivalent optimal basis closer to the lexicographically optimal one, where the chosen variable  $x_i$  is moved towards its lower bound—and hopefully becomes integer.

In our first heuristic, *Heur1*, when the objective function is degenerate we swap our focus to the candidate cut generating variable, i.e., the variable  $x_i$  to be perturbed is chosen as the most lex-significant fractional variable. The idea is that each new cut should guarantee a significant lex-decrease in the solution vector by either moving to a new vertex where the cut generating variables becomes integer, or else some other more lex-significant variables becomes fractional and can be cut.

A second perturbation heuristic, *Heur2*, can be designed along the following lines. Consider the addition of a single FGC and the subsequent tableau reoptimization performed by a standard dual simplex method. After the first pivot operation, the slack variable associated with the new cut goes to zero and leaves the basis, and it is unlikely that it will re-enter it in a subsequent step. This however turns out to be undesirable in the long run, since it increases the chances that the FGC generated in the next iterations will involve the slack variables of the previously-generated FGCs, and hence it favors the generation of cuts of higher rank and the propagation of their undesirable characteristics (density, numerical inaccuracy, etc.). By exploiting dual degeneracy, however, one could

try to select an equivalent optimal basis that includes the slack variables of the FGCs. This can be achieved by simply giving a small negative cost to the FGC slack variables.

Both the heuristics above involve the use of a small perturbation in the objective function coefficients, that however can produce numerical troubles that interfere with our study. So we handled perturbation in a way similar to that used in our implementation of the lexicographic dual simplex, that requires the solution of two LPs—one with the standard objective function, and the second with the second-level objective function and all nonbasic variables having nonzero reduced cost fixed at their bound.

## 5 Computational Results

Our set of pure ILP instances mainly comes from MIPLIB 2003 and MIPLIB 3; see Table 1. It is worth noting that, to our knowledge, even very small instances of these libraries (such as `stein15`, `bm23`, etc.) have never been solved by a pure cutting plane method based on FGC or GMI cuts read from the LP tableau.

Table 1. Our test bed

Problem	Cons	Vars	LP opt	Opt	Source
air04	823	8904	55535.44	56137	MIPLIB 3.0
air05	426	7195	25877.61	26374	MIPLIB 3.0
bm23	20	27	20.57	34	MIPLIB
cap6000	2176	6000	-2451537.33	-2451377	MIPLIB 3.0
hard_ks100	1	100	-227303.66	-226649	Single knapsack
hard_ks9	1	9	-20112.98	-19516	Single knapsack
krob200	200	19900	27347	27768	2 matching
l152lav	97	1989	4656.36	4722	MIPLIB
lin318	318	50403	38963.5	39266	2 matching
lseu	28	89	834.68	1120	MIPLIB
manna81	6480	3321	-13297	-13164	MIPLIB 3.0
mitre	2054	9958	114740.52	115155	MIPLIB 3.0
mzzv11	9499	10240	-22945.24	-21718	MIPLIB 3.0
mzzv42z	10460	11717	-21623	-20540	MIPLIB 3.0
p0033	16	33	2520.57	3089	MIPLIB
p0201	133	201	6875	7615	MIPLIB 3.0
p0548	176	548	315.29	8691	MIPLIB 3.0
p2756	755	2756	2688.75	3124	MIPLIB 3.0
pipex	2	48	773751.06	788263	MIPLIB
protfold	2112	1835	-41.96	-31	MIPLIB 3.0
sentoy	30	60	-7839.28	-7772	MIPLIB
seymour	4944	1372	403.85	423	MIPLIB 3.0
stein15	35	15	5	9	MIPLIB
stein27	118	27	13	18	MIPLIB 3.0
timtab	171	397	28694	764772	MIPLIB 3.0

Input data is assumed to be integer. All problems are preprocessed by adding an integer variable  $x_0$  that accounts for the original objective function, from which we can derive valid cuts, as Gomory's proof of convergence prescribes. Once a FGC is generated, we put it in its all-integer form in the space of the structural variables. In order to control round-off propagation, our FGC separator uses a threshold of 0.1 to test whether a coefficient is integer or not:

a coefficient with fractional part smaller than 0.1 is rounded to its nearest integer, whereas cuts with larger fractionalities are viewed as unreliable and hence discarded.

We carried out our experiments in a Intel Core 2 Q6600, 2.40GHz, with a time limit of 1 hour of CPU time and a memory limit of 2GB for each instance.

Our first set of experiments addressed the *single-cut* version of Gomory's algorithm. Actually, at each iteration we decided to generate *two* FGCs from the selected cut generating row—one from the tableau row itself, and one from the same row multiplied by -1.

The choice of the cut generation row in case of the lexicographic method is governed by the rule that prescribes the selection of the least-index variable. As to the other methods under comparison, the cut generation row is chosen with a random policy giving a higher probability of selecting the cut-generating variable from those with fractional part closer to 0.5 (alternative rules produced comparable results).

A very important implementation choice concerns the cut purging criterion. The lexicographic algorithm ensures the lexicographic improvement of the solution vector after each reoptimization, thus allowing one to remove cuts as soon as they become slack at the new optimum. As far as other methods are concerned, however, we can safely remove cuts only when the objective function improves. Indeed, if the objective function remains unchanged a removed cut can be generated again in a subsequent iteration, and the entire algorithm can loop—a situation that we actually encountered during our experiments. We therefore decided to remove the slack cuts only when it is mathematically correct, i.e. after a nonzero change in the objective function value, though this policy can lead to an out-of-memory status after a long stalling phase.

Table 2 compares results on the textbook implementation of Gomory's algorithm (TB) and the lexicographic one (Lex). Besides the percentage of closed gap (*ClGap*), we report 3 tightly correlated parameters to better measure the performance of each method. The first parameter is the cut coefficients size (*Coeff.*): large coefficients, besides increasing the likelihood of numerical errors, can be a symptom of cut ineffectiveness since they are required to represent very small angles in the space of structural variables. The second parameter is the determinant of the optimal basis (*Det.*). In a sense, the problem being all-integer, the determinant is a measure of the distance from an integer solution: a unit determinant implies an all-integer tableau and solution. Since any coefficient in the tableau can be expressed as a rational number whose denominator is the determinant of the current basis  $B$ , the smallest fractional part we could encounter in a tableau is  $1/\det(B)$ —weird tableau entries correspond to large determinants. However, our experiments showed that there are instances with huge determinants (e.g., `mitre`) but numerically quite stable. This is because the size of the determinant is only a weak bound on the degree of fractionality of the solution, as the large denominator can be – and fortunately often is – compensated by a large numerator. A more reliable indicator of numerical precision loss is our third parameter, the condition number  $\kappa$  of the optimal basis,

which gives a measure of the inaccuracy of the finite-precision representation of a solution  $x$  to the linear system  $Bx = b$ .

In the table, only the maximum value of the three indicators above during the run is reported. The first column reports one of the following exit-status codes: (O) integer optimum, (tL) time limit, (cL) limit of 100,000 cuts, (M) out of memory, (E) numerical errors (either no cuts passed the integrality check, or the problem became infeasible), and (1E) if one of the reoptimizations required by the lexicographic method failed for numerical reasons.

A possible failure of the lexicographic method arises when a strict lexicographic improvement is not reached because of numerical issues. In these situations we are no longer protected against the TB drawbacks and we can fail. Precisely, in `sentoy` (single-cut) we failed to improve lexicographically for 27 iterations, in `p0548` for 2597 iterations and in `timtab1-int` for 3 iterations. In multi-cut versions, we failed in `sentoy` for 5 iterations, `p0201` for 57 iterations, in `p0548` for 173 iterations, in `p2756` for 5 iterations, and in `timtab1-int` for 5 iterations.

Table 2 shows clearly that in most cases the TB version has huge coefficient sizes, determinants and condition numbers, while in Lex all these values remain relatively small along the entire run. Moreover, Lex could solve to proven optimality 9 of the 25 instances of our testbed—some of these instances being notoriously hard for pure cutting plane methods.

For illustration purposes, Figure 3 gives a representation of the trajectory of the LP optimal vertices to be cut (along with a plot of the basis determinant) when the textbook and the lexicographic methods are used for instance `stein15`. In Figures 3(a) and (b), the vertical axis represents the objective function value. As to the XY space, it is a projection of the original 15-dimensional variable space. The projection is obtained by using a standard procedure available e.g. in *MATLAB* (namely, multidimensional scaling [4]) with the aim of preserving the metric of the original 15-dimensional space as much as possible. In particular, the original Euclidean distances tend to be preserved, so points that look close one to each other in the figure are likely to be also close in the original space.

According to Figure 3(a), the textbook method concentrates on cutting points belonging to a small region. This behavior is in a sense a consequence of the efficiency of the underlying LP solver, that has no incentive in changing the LP solution once it becomes optimal with respect to the original objective function—the standard dual simplex will stop as soon as a feasible point (typically very close to the previous optimal vertex) is reached. As new degenerate vertices are created by the cuts themselves, the textbook method enters a feedback loop that is responsible for the exponential growth of the determinant of the current basis, as reported in Figure 3(d).

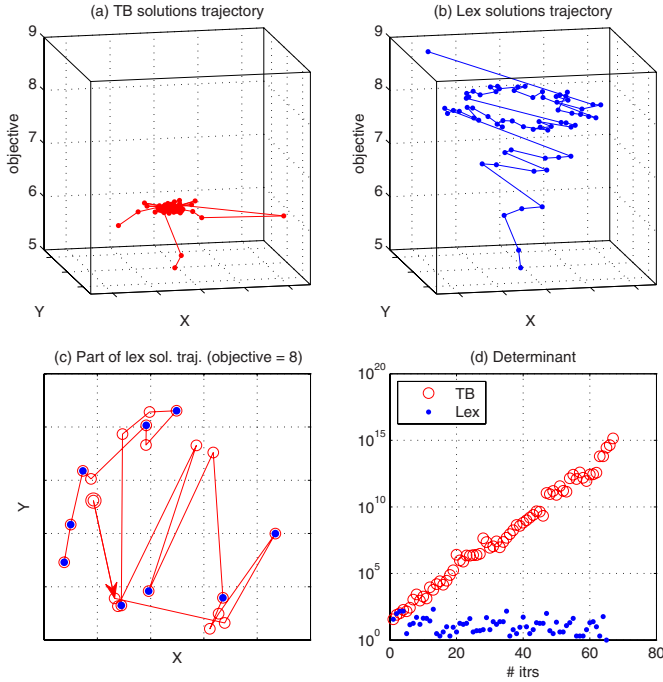
On the contrary, as shown in Figures 3(b), the lexicographic method prevents this by always moving the fractional vertex to be cut as far as possible (in the lex-sense) from the previous one. Note that, in principle, this property does not guarantee that there will be no numerical problems, but the method seems to be work pretty well in practice.



**Table 2.** Comparison between textbook and lexicographic implementation of Gomory’s algorithm (single-cut version)

Problem	Textbook						Lex							
	Itrs	Cuts	Time	ClGap	Coeff.	Det.	$\kappa$	Itrs	Cuts	Time	ClGap	Coeff.	Det.	$\kappa$
air04	tL 1197	2394	3602.85	4.42	1.4e+04	1.9e+18	1.8e+13	tL 371	742	3604.92	14.56	1e+04	5.2e+10	5.2e+18
air05	tL 2852	5704	3601.83	4.51	3.8e+05	1e+27	3e+15	tL 1012	2024	3604.92	22.44	4.8e+03	1.9e+10	3.5e+13
bm23	M 5370	5628	2733.43	18.09	1e+15	3.6e+32	2.7e+18	O 713	1426	2.40	100.00	2.4e+02	4.7e+10	1e+10
cap0000	tL 1666	3329	3603.51	8.47	1.5e+10	1.2e+21	9.2e+21	tL 594	1188	3604.17	14.08	2.5e+06	5.1e+09	4.8e+15
hard_ks100	O 3134	6237	1586.82	100.00	8.7e+05	4.1e+08	1.4e+13	O 214	428	1.10	100.00	6.7e+05	6.3e+05	1.5e+14
hard_ks9	O 1058	2107	2.74	100.00	8.3e+14	6.4e+21	1.8e+28	O 889	1778	0.74	100.00	4.8e+04	5.5e+06	1.4e+10
krob200	O 281	532	20.81	100.00	1.2e+02	4.9e+07	3.2e+08	tL 666	1280	3606.81	86.70	2.3e+03	8.6e+07	1e+17
ll52lav	tL 3412	6824	3602.10	13.16	6.1e+03	2.7e+09	1.3e+12	O 1152	2278	297.83	100.00	1.7e+04	1.9e+06	5.9e+11
lin318	O 1667	3289	1155.32	100.00	1.6e+03	1.1e+12	5.4e+11	tL 200	346	3618.80	57.69	1.2e+02	1.1e+12	2e+14
lseu	E 4115	8201	671.27	60.75	5.2e+14	7e+31	8.5e+30	O 13541	27068	70.74	100.00	2.4e+04	5.2e+18	2.7e+13
manna81	O 423	423	1338.29	100.00	1	4.9e+173	4.8e+06	81	81	3617.72	9.77	1	1.6e+178	4.8e+06
mitre	tL 6445	12882	3601.11	59.95	1.5e+08	Inf	4.2e+18	tL 77	154	3621.68	2.05	3.2e+03	Inf	1.9e+16
mzzv11	tL 203	406	3618.53	8.98	4.8e+02	6.4e+57	3.4e+12	tL 52	104	3746.52	7.68	61	4.3e+48	8.7e+12
mzzv42z	tL 195	390	3617.33	2.77	8.5e+02	8.3e+53	9.6e+12	tL 25	50	3810.13	5.45	37	7.9e+37	3.7e+14
p0033	E 2919	5824	1054.55	71.15	5.8e+14	1.8e+29	1.1e+31	O 1961	3832	4.22	100.00	2.3e+03	3.5e+17	2.9e+16
p0201	tL 14521	29036	3601.16	13.92	2.6e+11	7.9e+36	7.4e+25	IE 29950	58845	792.35	67.57	2.3e+05	1.2e+22	2.7e+16
p0548z	tL 19279	38446	3601.41	50.11	2.4e+12	Inf	3.4e+28	tL 29199	58216	3603.34	0.03	3.4e+06	2.7e+190	2.4e+16
p2756	tL 5834	11560	3601.75	78.63	5.5e+12	Inf	1e+27	IE 1787	3574	2957.59	0.52	9.1e+02	1.1e+278	5.4e+15
pipeX	E 4719	9408	24.84	36.26	7.5e+14	8.1e+26	3.1e+27	cL 50000	99390	146.86	42.43	1.1e+05	3e+10	3.4e+21
protfold	tL 68	136	3811.43	8.76	5.2e+10	Inf	2.1e+17	tL 299	598	3606.09	45.26	30	1.3e+30	2.1e+07
sentoy	E 830	1639	15.28	3.39	4.3e+14	2.1e+37	3e+28	O 5771	11541	47.88	100.00	6.5e+04	4.6e+34	1.5e+16
seymour	tL 124	246	3768.72	6.01	4.7e+08	5e+56	9.1e+20	tL 94	182	3618.15	11.23	15	1.2e+19	1.6e+08
stein15	E 173	313	2.29	12.39	2.1e+16	Inf	2.5e+20	O 65	121	0.18	100.00	17	3.6e+02	1.8e+11
stein27	E 208	365	3.59	0	1.5e+16	Inf	7e+21	O 4298	8301	45.23	100.00	7.2e+02	8.9e+04	1.2e+06
timtab1-int	tL 10784	21501	3603.58	11.94	1.7e+13	Inf	1.2e+27	cL 50000	99887	3124.58	4.00	1.6e+07	6e+250	1.5e+21





**Fig. 3.** Problem stein15 (single cut). (a)-(b) Solution trajectories for TB and Lex, resp.; (c) Lower dimensional representation of the the Lex solution trajectory; the filled circles are lexicographic optima used for cut separation; their immediate next circles are optima given by the black-box dual-simplex solver, whereas the other points correspond to the equivalent solutions visited during lexicographic reoptimization; the double circle highlights the trajectory starting point. (d) Growth of determinants in TB and Lex (logarithmic scale).

Finally, Figure 3(c) offers a closer look at the effect of lexicographic reoptimization. Recall that our implementation of the lexicographic dual simplex method involves a sequence of reoptimizations, each of which produces an alternative optimal vertex possibly different from the previous one. As a result, between two consecutive cuts our method internally traces a trajectory of equivalent solutions, hence in the trajectory plotted in Figure 3(b) we can distinguish between two contributions to the movement of  $x^*$  after the addition of a new cut: the one due to the black-box optimizer, and the one due to lex-reoptimization. Figure 3(c) concentrates on the slice objective=8 of the Lex trajectory. Each lexicographic optimal vertex used for cut separation is depicted as a filled circle. The immediate next point in the trajectory is the optimal vertex found by the standard black-box dual simplex, whereas the next ones are those contributed by the lexicographic reoptimization. The figure shows that lexicographic reoptimization has a significant effect in moving the points to be cut, that in some cases are very far from those returned by the black-box dual simplex.

Table 3. Comparison between textbook and lexicographic implementation of Gomory’s algorithm (multi-cut version)

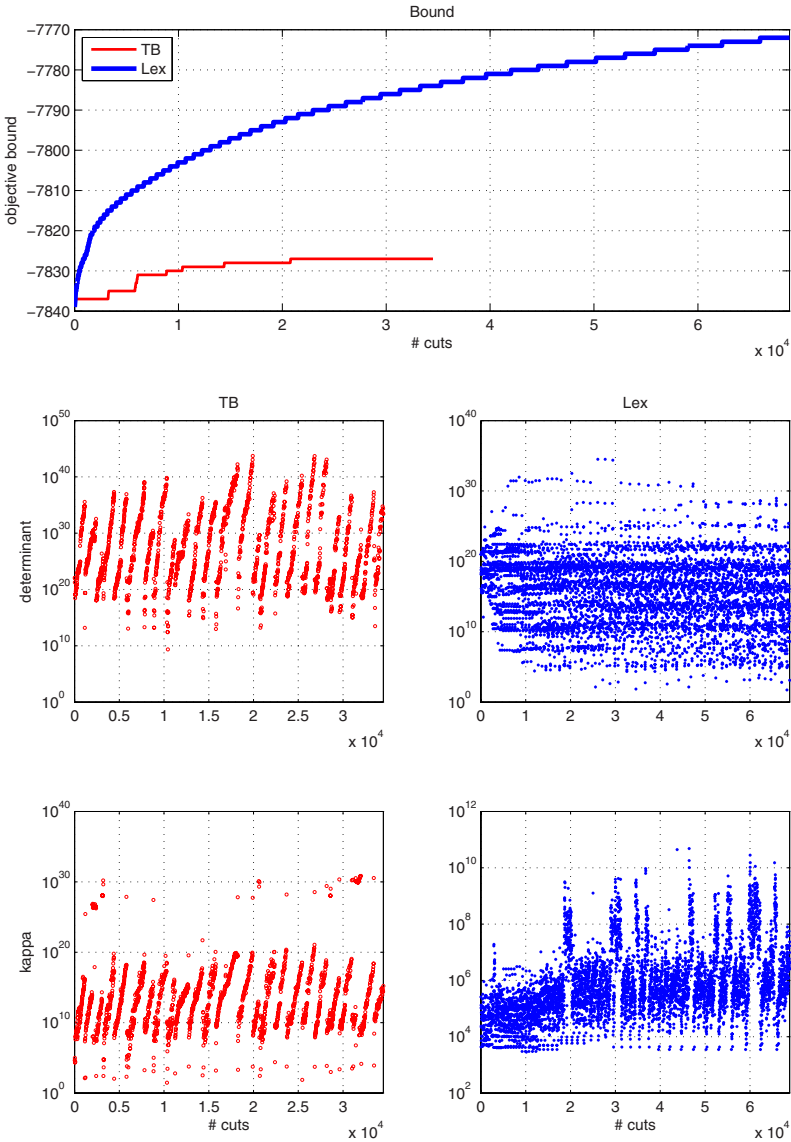
Problem	Textbook				Lex											
	Itrs	Cuts	Time	ClGap	Coeff.	Det.	$\kappa$	Itrs	Cuts	Time	ClGap	Coeff.	Det.	$\kappa$		
air04	M	35	11912	1209.41	11.23	2.4e+03	6.9e+11	9.3e+09	tL	150	47995	3606.61	<b>19.21</b>	7.3e+03	2.2e+09	3.9e+11
air05	M	30	9648	955.09	5.11	7.1e+02	4.2e+11	4.9e+09	tL	261	76263	3598.20	<b>14.00</b>	1.5e+03	4.9e+10	9.6e+10
bm23	E	144	2168	9.33	5.28	2.3e+15	5.5e+30	8.7e+24	O	659	8298	1.67	<b>100.00</b>	3.7e+02	4.6e+10	3e+14
cap6000	tL	979	11693	3614.59	10.34	1.4e+09	7e+15	4.3e+20	E	966	6769	2110.12	<b>24.66</b>	4.5e+06	5.1e+09	1.6e+15
hard_ks100	tL	1950	10770	3603.03	<b>100.00</b>	5e+04	2.2e+05	1.5e+12	O	98	439	0.63	<b>100.00</b>	5e+05	1.5e+04	2.8e+09
hard_ks9	O	269	1678	0.30	<b>100.00</b>	7.8e+04	1.7e+05	6.9e+09	O	139	614	0.12	<b>100.00</b>	3e+04	3.6e+04	2.9e+10
krob200	O	44	2017	153.92	<b>100.00</b>	1.6e+02	1e+06	9.4e+07	O	17	282	22.18	<b>100.00</b>	7.2	1e+06	3.7e+04
l152lav	M	525	31177	2715.78	40.59	5.3e+03	8e+08	1.6e+10	O	681	22364	206.82	<b>100.00</b>	1.8e+04	6.7e+05	2.4e+10
lin318	O	15	250	12.82	<b>100.00</b>	7	3.4e+07	9.7e+05	O	19	416	124.27	<b>100.00</b>	17	3.2e+07	1.8e+08
lseu	E	149	3710	34.35	44.58	1.7e+14	9.8e+33	2e+19	O	1330	18925	6.62	<b>100.00</b>	9.6e+03	1.2e+14	2.6e+14
manna81	O	1	270	8.88	<b>100.00</b>	1	5.2e+92	3.7e+06	O	2	280	29.58	<b>100.00</b>	1	1.7e+97	3.7e+06
mitre	M	94	30016	1262.24	85.52	8.9e+08	Inf	4.2e+20	tL	232	20972	3619.30	<b>97.83</b>	6.2e+06	Inf	9.1e+13
mzzv11	M	33	17936	1735.42	38.56	4.9e+02	Inf	2.6e+11	tL	61	25542	3739.41	<b>40.60</b>	1.8e+02	2.8e+51	8.5e+11
mzzv42z	M	62	14664	1515.50	<b>33.80</b>	3e+06	Inf	6.5e+14	tL	61	14830	3616.85	25.50	8.5e+02	1e+38	8.7e+12
p0033	M	1529	23328	2493.35	81.53	1.8e+14	1.1e+34	2.2e+24	O	87	1085	0.14	<b>100.00</b>	4.5e+02	1.6e+13	6e+14
p0201	tL	1332	55408	3602.21	19.32	2e+12	2e+34	2.5e+26	E	27574	629004	1119.71	<b>74.32</b>	2.3e+05	3.6e+22	7.4e+11
p0548	M	825	35944	2223.35	<b>48.98</b>	1.3e+09	2.4e+137	2e+24	E	461	27067	131.69	47.50	4.8e+06	7.1e+135	5.5e+14
p2756	M	740	19925	2423.48	78.86	4.5e+12	Inf	1.3e+27	E	642	14645	509.72	<b>79.09</b>	2.4e+05	Inf	1.1e+13
pipex	M	2929	49391	1921.78	51.25	1.3e+14	3.3e+27	4e+28	O	5000	488285	188.11	<b>74.18</b>	2.8e+05	8.5e+11	1e+14
profitfold	M	7	5526	1058.84	8.76	4.6e+06	Inf	5.1e+13	tL	159	38714	3607.08	<b>45.26</b>	30	1.5e+32	1e+07
seymour	M	1953	34503	2586.38	18.25	5.1e+14	5.3e+43	7e+30	O	5331	68827	25.85	<b>100.00</b>	7.6e+04	3.2e+34	3.9e+14
stein15	tL	18	11748	7517.40	21.67	3.3e+08	1.3e+159	1.4e+16	tL	87	48339	3610.62	<b>26.89</b>	78	1e+23	3.6e+07
stein27	E	116	3265	9.50	20.92	2.5e+15	2.4e+26	3.2e+28	O	64	676	0.14	<b>100.00</b>	15	2e+02	2.3e+06
timtab1-int	M	231	73188	585.72	0.00	1.7e+15	2.8e+36	8.3e+18	O	3175	37180	19.72	<b>100.00</b>	7.5e+02	1.9e+04	1.9e+05
					23.59	1.6e+11	Inf	2.1e+21	cL	3165	1000191	841.56	<b>50.76</b>	3.5e+06	Inf	1.7e+16

Table 4. The two heuristics compared (single-cut version)

Problem	Heur1					Heur2								
	Itrs	Cuts	Time	CI/Gap	Coeff.	Det.	$\kappa$	Itrs	Cuts	Time	CI/Gap	Coeff.	Det.	$\kappa$
air04	tL 1886	3772	3602.45	6.24	7.9e+04	6.5e+27	1.5e+15	tL 1463	2926	3601.57	12.06	1.3e+07	7e+15	5.2e+18
air05	tL 1355	2710	3630.37	4.11	1.1e+10	1.6e+51	4.1e+21	tL 1338	2676	3603.34	4.51	1.1e+05	8.6e+26	3.5e+13
bm23	tL 5760	6111	3602.69	25.54	1.7e+15	2.2e+29	1.7e+20	tL 3938	7876	3602.32	25.54	1.1e+04	2.1e+15	1e+10
cap6000	tL 2024	4048	3601.36	7.85	2.8e+09	6.5e+20	9.6e+20	tL 1379	2757	3605.44	8.47	4.5e+06	3.8e+13	4.8e+15
hard_ks100	O 2212	4403	742.51	100.00	4.9e+11	3.3e+21	1.7e+24	O 627	1251	11.10	100.00	6.4e+05	5.4e+06	1.5e+14
hard_ks9	O 839	1674	0.67	100.00	1.6e+05	1.6e+07	2.5e+11	O 162	322	0.10	100.00	5.4e+05	5.1e+05	1.4e+10
krob200	tL 3009	5978	3602.40	92.40	1.1e+03	1e+16	8.2e+11	M 3051	6066	3544.19	97.15	7.5e+05	1.3e+13	1e+17
l152lav	tL 5910	11813	3601.22	46.68	2.7e+05	4.6e+11	6.4e+13	tL 2071	4140	3604.68	32.97	2e+04	8.2e+16	5.9e+11
lin318	M 1699	3360	2760.11	66.94	5.6e+03	1.2e+14	6.1e+12	M 1002	1966	1203.28	88.43	2.1e+04	2.1e+13	2e+14
lseu	tL 17687	35367	3601.67	69.16	7.3e+11	7.7e+32	1.8e+24	tL 3805	7608	3601.04	47.78	2.5e+05	1.6e+24	2.7e+13
manma81	O 425	425	1329.69	100.00	1	4.9e+173	4.8e+06	O 423	423	1328.05	100.00	1	4.9e+173	4.8e+06
mitre	tL 6003	12003	3601.02	47.89	2.3e+05	Inf	1.6e+14	tL 5973	11934	3601.13	75.15	3e+06	Inf	1.9e+16
mzvv11	tL 69	138	3623.00	4.44	1.7e+02	Inf	5.4e+12	tL 267	532	3612.87	13.45	8.2e+02	2.8e+54	8.7e+12
mzvv42z	tL 125	250	3667.32	5.72	3.2e+02	1.2e+65	2.9e+12	tL 247	494	3601.63	8.96	9.3e+03	1.9e+54	3.7e+14
p0033	O 20667	41287	486.85	100.00	3.3e+08	7.6e+20	1.8e+19	tL 14143	28267	3603.52	94.55	1.3e+08	4e+17	2.9e+16
p0201	tL 10756	21510	3602.02	21.22	3.6e+09	4e+32	8.7e+21	tL 3689	7378	3601.73	17.16	7.6e+06	3.3e+21	2.7e+16
p0548	tL 22096	44063	3601.44	50.61	1.5e+06	4.7e+190	1.4e+18	tL 9523	19007	3603.13	43.19	5.8e+06	1e+190	2.4e+16
p2756	tL 4147	8191	3601.76	78.63	5.6e+10	Inf	1.1e+21	tL 4623	9182	3604.52	77.25	3.8e+04	Inf	5.4e+15
pipex	E 50000	99985	650.65	47.51	1.4e+08	1.8e+19	1.2e+17	tL 26620	53233	3601.96	40.68	1.4e+11	2e+19	3.4e+21
profold	tL 366	732	3617.67	17.88	4.6	9.6e+37	9.8e+09	tL 435	870	3610.08	8.76	1.5	2.4e+28	2.1e+07
sentoy	E 1184	2357	116.95	7.85	3.9e+14	2.5e+38	1e+29	tL 6951	13900	3602.91	22.71	3.3e+06	8.6e+30	1.5e+16
seymour	tL 201	401	3622.41	6.01	9.1e+02	4.6e+26	1.8e+12	tL 275	548	3615.12	11.23	17	4.9e+17	1.6e+08
stein15	tL 5592	11145	3601.33	75.00	1.2e+03	1.5e+11	1e+09	tL 4148	8293	3602.46	50.00	4.9e+04	1.3e+15	1.8e+11
stein27	tL 3798	7595	3601.57	0.00	1.4e+02	1.1e+11	1.6e+08	tL 3645	7290	3601.92	0.00	6	1.9e+05	1.2e+06
timtab1-int	cL 50000	99997	1676.40	16.02	1.3e+07	Inf	4.3e+18	tL 21474	42946	3601.19	14.50	1.1e+09	Inf	1.5e+21

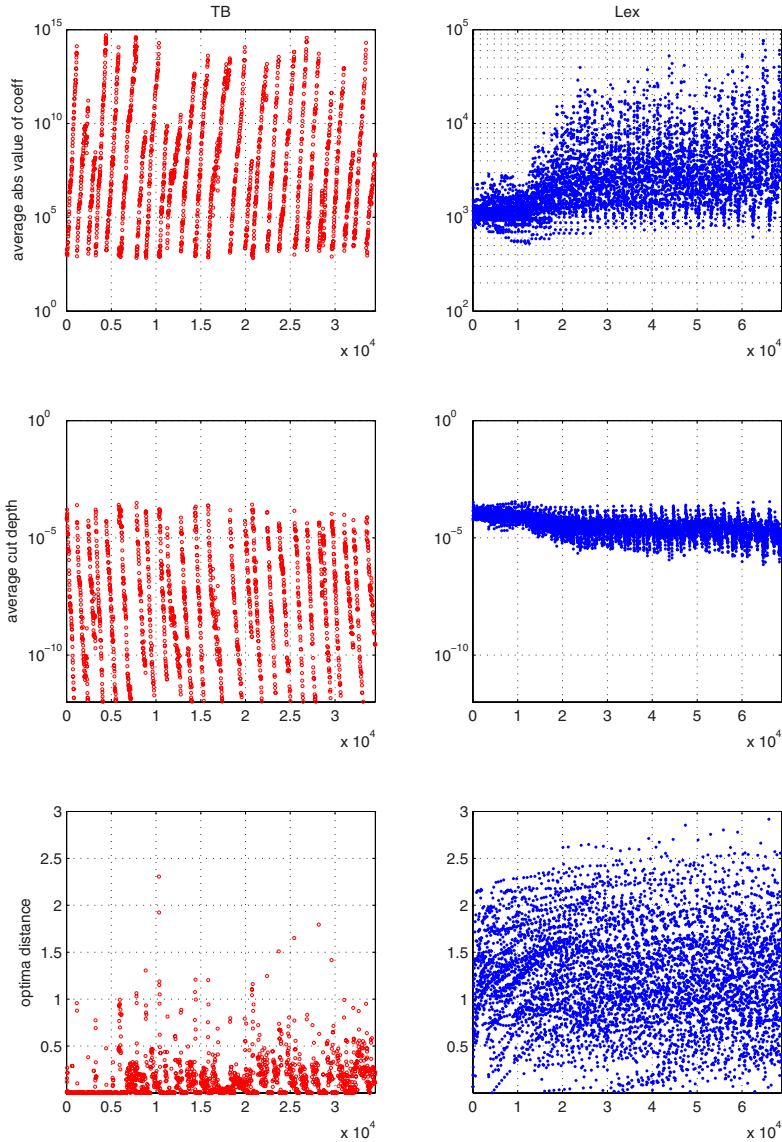
Table 5. The two heuristics compared (multi-cut version)

Problem	Heur1					Heur2								
	Itrs	Cuts	Time	ClGap	Coeff.	Det.	$\kappa$	Itrs	Cuts	Time	ClGap	Coeff.	Det.	$\kappa$
air04	M 30	10250	1287.10	10.40	1.4e+04	2.1e+10	1.8e+12	M 28	9075	511.50	9.73	2.6e+03	2.1e+08	3.9e+11
air05	M 49	15805	1365.48	6.33	3.1e+04	4.1e+24	2.1e+11	M 26	8288	667.99	5.11	1.8e+03	3.5e+10	9.6e+10
bm23	M 1819	8403	3112.35	18.09	5e+14	1.7e+29	6.7e+22	M 631	13800	1933.80	25.54	4e+06	3.5e+23	3e+14
cap6000	tL 484	7002	3605.86	7.23	3.5e+08	4e+16	3.1e+17	M 559	7511	2498.47	9.72	1.5e+08	8e+14	1.6e+15
hard_ks100	tL 150	672	1.24	100.00	1.2e+04	3.3e+04	1.8e+10	O 810	1642	28.45	100.00	3.5e+03	3e+03	2.8e+09
hard_ks9	O 251	1641	0.33	100.00	5.5e+04	2.6e+05	8.3e+10	O 252	2010	1.18	100.00	4.9e+05	2.7e+08	2.9e+10
krob200	O 12	108	3.15	100.00	1.2	1e+06	1.4e+05	O 9	81	2.27	100.00	1.2	1e+06	3.7e+04
1152lav	tL 627	30047	3639.62	46.68	1.4e+09	2.3e+24	5.4e+15	M 106	9889	1034.03	31.44	6.7e+03	1.1e+10	2.4e+10
lin318	O 42	2415	563.91	100.00	1e+02	4.7e+09	5.6e+08	O 35	1503	238.84	100.00	55	7.6e+08	1.8e+08
lseu	M 2604	54498	3388.00	70.21	2.9e+12	2.1e+32	4.8e+19	M 542	16610	1433.12	48.48	8e+06	5.4e+23	2.6e+14
manua81	O 1	270	8.94	100.00	1	5.2e+92	3.7e+06	O 1	270	8.83	100.00	1	5.2e+92	3.7e+06
mitre	tL 148	47954	3651.05	88.90	5.4e+07	Inf	1.1e+17	tL 192	27786	3624.63	91.56	3.4e+04	Inf	9.1e+13
mzvv11	tL 14	10452	5544.73	29.27	8.9e+03	4.9e+73	4.4e+15	M 40	15727	1699.51	34.16	3.5e+02	2.9e+43	8.5e+11
mzvv42z	M 18	6818	2613.82	18.28	2.6e+04	1.2e+52	3.4e+13	M 74	17099	2423.77	28.44	1.5e+03	1.4e+41	8.7e+12
p0033	O 40	460	0.06	100.00	2.5e+02	1.5e+13	4.1e+06	M 2060	37057	3190.24	91.03	3e+07	4.9e+16	6e+14
p0201	tL 2087	92066	3612.42	51.22	1.1e+09	1.8e+26	8.8e+20	M 756	26159	2476.90	37.57	6.9e+04	1.6e+15	7.4e+11
p0548	tL 1401	90042	3601.77	53.75	4.7e+07	2.4e+137	1.9e+16	M 358	24003	1612.11	46.09	2.7e+04	5.1e+133	5.5e+14
p2756	tL 420	17108	3613.50	78.86	8e+12	Inf	3.1e+27	M 283	14402	1852.05	78.40	4.8e+04	Inf	1.1e+13
pipex	E 50000	552448	752.66	55.44	5.3e+09	1e+20	1.5e+20	M 1530	30972	1608.28	48.98	2.2e+07	1.9e+16	1e+14
protfold	tL 19	7886	3775.30	27.01	6.9	6.5e+28	1.2e+09	tL 23	8488	3674.48	8.76	1.4	2.6e+24	1e+07
scenty	E 118	1712	5.06	4.88	5.1e+14	1.3e+36	1.1e+20	M 1064	20822	2111.59	22.71	7.9e+05	3.2e+29	3.9e+14
scymour	M 19	12084	3081.48	21.67	1.3e+02	1.1e+31	1.6e+11	M 30	16270	3494.90	26.89	8.8	2.9e+17	3.6e+07
stein15	M 985	20707	2133.91	75.00	1.9e+03	1.1e+11	1.5e+09	M 505	13838	1585.78	50.00	44	3.7e+05	2.3e+06
stein27	M 283	13587	937.55	0.00	39	9.8e+06	4.2e+06	M 274	13586	936.43	0.00	4.7	1.8e+05	1.9e+05
timtab1-int	M 383	119945	288.72	37.10	6.2e+06	3.9e+216	3.1e+13	M 688	197581	733.33	43.08	1.4e+07	8.1e+198	1.7e+16



**Fig. 4.** Comparison between the textbook and lexicographic implementations of multi-cut Gomory’s algorithm on `sentoy`

To support the interpretation above even further, we performed the experiment of just restarting the LP solver from scratch after having generated the FGCs, so that it is more likely that a “substantially different” optimal solution is found. This small change had a significant impact on the performance of the textbook method (though not comparable to that derived from the use of the



**Fig. 5.** Comparison between the textbook and lexicographic implementations of multi-cut Gomory’s algorithm on *sentoy*

lexicographic method), showing the importance of breaking the correlation of the optimal LP bases.

Table 4 reports the results of our two heuristics, *Heur1* and *Heur2*. A comparison with the previous table shows that both heuristics are effective in controlling

the coefficient size, determinant, and condition number. The average closed gap is significantly better than in TB, but clearly worse than in Lex.

A second set of experiments was carried out on the *multi-cut* version of Gomory’s algorithm, where cuts are generated in rounds. To be specific, after each LP reoptimization we consider all the tableau rows with fractional basic variable, and generate two FGCs from each row—one from the row itself, and one from the same row multiplied by  $-1$ .

According to Table 3, the multi-cut version of Lex performed even better than in the single-cut mode: in 13 out of the 26 instances the method reached the optimum. Figures 4 and 5 give some illustrative plots for instance `sentoy`. The figures clearly show the typical degenerate behavior of TB, with instable phases of rapid growth of determinant/coefficients/ $\kappa$  exploring small space regions with shallow cuts. It is worth observing the striking difference in the plots of the *average cut depth*, computed as the geometric distance of the cut from the separated vertex, averaged over all the cuts in a round. Even more interesting, the TB and Lex have a completely different behavior as far as the *optima distance* (computed as the Euclidean distance between two consecutive fractional vertices to be cut) is concerned. As a matter of fact, as already shown by Figure 3, lexicographic reoptimization is quite successful in amplifying the dynamic (and diversity) of the fractional solutions.

## 6 Conclusions and Future Work

Pure cutting plane algorithms have been found not to work in practice because of numerical problems due to the cuts becoming increasingly parallel (a phenomenon accompanied by dual degeneracy), increasing determinant size and condition number, etc. For these reasons, cutting planes are in practice used in cut-and-branch or branch-and-cut mode.

In this paper we have discussed an implementation of the lexicographic version of Gomory’s fractional cutting plane method and of two heuristics mimicking the latter one. In computational testing on a battery of MIPLIB problems, we compared the performance of these variants with that of the standard Gomory algorithm, both in the single-cut and in the multi-cut (rounds of cuts) version, and showed that they provide a radical improvement over the standard procedure. In particular, we reported the exact solution of ILP instances from MIPLIB such as `stein15`, `stein27`, and `bm23`, for which the standard Gomory cutting plane algorithm is not able to close more than a tiny fraction of the integrality gap.

The significance of these result suggests that the lexicographic approach can be applied to any cutting plane algorithm—no matter what kind of cuts you add in the step that generates cuts, you may reoptimize using the lexicographic dual simplex method so as to break dual degeneracy in favor of “cleaner” LP bases associated with better primal vertices to cut. We plan to investigate this topic in the near future.

## References

1. Arthur, J.L., Ravindran, A.: PAGP, a partitioning algorithm for (linear) goal programming problems. *ACM Trans. Math. Softw.* 6(3), 378–386 (1980)
2. Balas, E., Ceria, S., Cornuéjols, G., Natraj, N.: Gomory cuts revisited. *Operations Research Letters* 19, 1–9 (1996)
3. Balinski, M.L., Tucker, A.W.: Duality theory of linear programs: A constructive approach with applications. *SIAM Review* 11(3), 347–377 (1969)
4. Borg, I., Groenen, P.J.F.: *Modern Multidimensional Scaling: Theory and Applications*. Springer, Heidelberg (2005)
5. Balas, E., Saxena, A.: *Optimizing over the split closure*. *Mathematical Programming* (2006)
6. Fischetti, M., Lodi, A.: *Optimizing over the first Chvátal closure*. *Mathematical Programming B* 110(1), 3–20 (2007)
7. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Society* 64, 275–278 (1958)
8. Gomory, R.E.: *An algorithm for the mixed integer problem*. Technical Report RM-2597, The RAND Cooperation (1960)
9. Gomory, R.E.: *An algorithm for integer solutions to linear programming*. In: Graves, R.L., Wolfe, P. (eds.) *Recent Advances in Mathematical Programming*, pp. 269–302. McGraw-Hill, New York (1963)
10. Tamiz, M., Jones, D.F., El-Darzi, E.: A review of goal programming and its applications. *Annals of Operations Research* (1), 39–53 (1995)



# The Mixing Set with Divisible Capacities<sup>\*</sup>

Michele Conforti<sup>1</sup>, Marco Di Summa<sup>1</sup>, and Laurence A. Wolsey<sup>2</sup>

<sup>1</sup> Dipartimento di Matematica Pura ed Applicata, Università degli Studi di Padova,  
Via Trieste 63, 35121 Padova, Italy

{conforti,mdsumma}@math.unipd.it

<sup>2</sup> Center for Operations Research and Econometrics (CORE), Université catholique  
de Louvain, 34, Voie du Roman Pays, 1348 Louvain-la-Neuve, Belgium  
laurence.wolsey@uclouvain.be

**Abstract.** Given rational numbers  $C_0, \dots, C_m$  and  $b_0, \dots, b_m$ , the mixing set with arbitrary capacities is the mixed-integer set defined by conditions

$$\begin{aligned} s + C_t z_t &\geq b_t, & 0 \leq t \leq m, \\ s &\geq 0, \\ z_t &\text{integer}, & 0 \leq t \leq m. \end{aligned}$$

Such a set has applications in lot-sizing problems. We study the special case of divisible capacities, i.e.  $C_t/C_{t-1}$  is a positive integer for  $1 \leq t \leq m$ . Under this assumption, we give an extended formulation for the convex hull of the above set that uses a quadratic number of variables and constraints.

**Keywords:** Mixed-integer programming, compact extended formulations, mixing sets.

## 1 Introduction

Given rational numbers  $C_0, \dots, C_m$  and  $b_0, \dots, b_m$ , the mixing set with arbitrary capacities is the mixed-integer set defined by conditions

$$s + C_t z_t \geq b_t, \quad 0 \leq t \leq m, \tag{1}$$

$$s \geq 0, \tag{2}$$

$$z_t \text{ integer}, \quad 0 \leq t \leq m. \tag{3}$$

The above set generalizes the *mixing set*, which is a set of the type (1)–(3) with  $C_t = 1$  for all  $0 \leq t \leq m$ . The mixing set, which was introduced and studied by Günlük and Pochet [9] and further investigated by Miller and Wolsey [12], has played an important role in studying production planning problems (in particular lot-sizing [17]).

---

<sup>\*</sup> This work was partly carried out within the framework of ADONET, a European network in Algorithmic Discrete Optimization, contract no. MRTN-CT-2003-504438.

When the values of the capacities  $C_t$  are arbitrary, (1)–(3) constitutes a relaxation of lot-sizing problems where different batch sizes or velocities of the machines are allowed. Giving a linear inequality description of the convex hull of such a set seems to be difficult and indeed it is not known whether linear optimization over (1)–(3) can be carried out in polynomial time.

We consider here the special case of a set defined by (1)–(3) where the capacities form a sequence of divisible numbers: that is,  $C_t/C_{t-1}$  is a positive integer for  $1 \leq t \leq m$ . We call such a set the *mixing set with divisible capacities* and we denote it by  $DIV$ . Our main result is a compact extended formulation for the polyhedron  $\text{conv}(DIV)$ , the convex hull of  $DIV$ .

Here we use the following terminology. A *formulation* of a polyhedron  $P$  (in its original space) is a description of  $P$  as the intersection of a finite number of half-spaces. So it consists of a system of linear inequalities  $Cx \geq d$  such that  $P = \{x : Cx \geq d\}$ . A formulation of  $P$  is *extended* whenever it gives a polyhedral description of the type  $Q = \{(x, \mu) : Ax + B\mu \geq d\}$  in a space that uses variables  $(x, \mu)$  and includes the original  $x$ -space, so that  $P$  is the projection of  $Q$  onto the  $x$ -space.

If  $P$  is the convex hull of a mixed-integer set (such as the convex hull of the set defined by (1)–(3)), we say that a formulation is *compact* if its size (i.e. the number of inequalities and variables of the system defining  $P$  or  $Q$  as above) is bounded by a polynomial function of the description of the mixed-integer set (in our case the size of the system (1)–(2)).

The assumption of divisibility of the coefficients was exploited by several authors to tackle integer sets that are otherwise untractable, such as integer knapsack problems. Under the divisibility assumption, Marcotte [11] gave a simple formulation of the integer knapsack set without upper bounds on the variables. Pochet and Wolsey [16] studied the same set where the knapsack inequality is of the “ $\geq$ ” type. Pochet and Weismantel [13] provided a linear inequality description of the knapsack set where all variables are bounded. Other hard problems studied under the assumption of divisibility of the coefficients include network design [14], lot-sizing problems [4] and the integer Carathéodory property for rational cones [10].

The mixing set with divisible capacities  $DIV$  was studied recently by Zhao and de Farias [20], who gave a polynomial-time algorithm to optimize a linear function over  $DIV$  (see also Di Summa [6]).

A formulation of the polyhedron  $\text{conv}(DIV)$  either in the original space or in an extended space was not known for the general case and such a formulation does not seem to be easily obtainable by applying known techniques for constructing compact extended formulations, such as taking unions of polyhedra [14] or enumeration of fractional parts [12,3,18,19].

A formulation of  $\text{conv}(DIV)$  was only known for some special cases. For the set  $DIV$  with  $C_t = 1$  for  $0 \leq t \leq m$  (i.e. the mixing set), a linear inequality description of the convex hull in the original space was given by Günlük and Pochet [9] and a compact extended formulation was obtained by Miller and Wolsey [12]. For the set  $DIV$  with only two distinct values of the capacities,

Van Vyve [18] and Constantino, Miller and Van Vyve [5] gave a linear inequality description of the convex hull of the set both in the original space and in an extended space. Zhao and de Farias [20] gave a linear inequality formulation of  $\text{conv}(DIV)$  in its original space under some special assumptions on the parameters  $C_0, \dots, C_m$  and  $b_0, \dots, b_m$ .

Since a polynomial-time algorithm for the set  $DIV$  was already known, one might wonder why we are interested in giving a polyhedral description of  $DIV$ . However recall that mixed-integer sets of the type (1)–(3) appear as substructures in multi-item lot-sizing problems, thus a linear inequality description of  $\text{conv}(DIV)$  leads to strong formulations for such problems.

In order to study the set  $DIV$ , we rewrite (1)–(3) in a slightly different form, as we need to have  $C_t \neq C_{t'}$  for  $t \neq t'$ . In other words, we group together the inequalities (1) associated with the same capacity  $C_t$  and write the set  $DIV$  as follows:

$$s + C_k z_t \geq b_t, \quad t \in I_k, 0 \leq k \leq n, \tag{4}$$

$$s \geq 0, \tag{5}$$

$$z_t \text{ integer}, \quad t \in I_0 \cup \dots \cup I_n, \tag{6}$$

where  $I_0, \dots, I_n$  are pairwise disjoint sets of indices and  $C_k/C_{k-1}$  is an integer greater than one for  $1 \leq k \leq n$ .

The main idea of our approach to construct a compact extended formulation for  $\text{conv}(DIV)$  can be summarized as follows: We consider the following expansion of  $s$ :

$$s = \alpha_0(s) + \sum_{j=1}^{n+1} \alpha_j(s)C_{j-1},$$

where  $0 \leq \alpha_j(x) < \frac{C_j}{C_{j-1}}$  for  $1 \leq j \leq n$ , and  $0 \leq \alpha_0(x) < C_0$ . Furthermore  $\alpha_j(x)$  is an integer for  $1 \leq j \leq n+1$ . We show that for fixed  $j$ , the number of possible values that  $\alpha_j(s)$  can take over the set of vertices of  $\text{conv}(DIV)$  is bounded by a linear function of the number of constraints (1). To each of these possible values (say  $v$ ), we associate an indicator variable that takes value 1 if  $\alpha_j(s) = v$  and 0 otherwise. These indicator variables are the important additional variables of our compact extended formulation.

## 2 Expansion of a Number

Our arguments are based on the following expansion of a real number  $x$ :

$$x = \alpha_0(x) + \sum_{j=1}^{n+1} \alpha_j(x)C_{j-1}, \tag{7}$$

where  $0 \leq \alpha_j(x) < \frac{C_j}{C_{j-1}}$  for  $1 \leq j \leq n$ , and  $0 \leq \alpha_0(x) < C_0$ . Furthermore  $\alpha_j(x)$  is an integer for  $1 \leq j \leq n+1$ . Note that this expansion is unique. If we let

$$f_0(x) = \alpha_0(x), \quad f_k(x) = f_0(x) + \sum_{j=1}^k \alpha_j(x)C_{j-1} \quad \text{for } 1 \leq k \leq n,$$

we have that

$$x = f_k(x) + \sum_{j=k+1}^{n+1} \alpha_j(x)C_{j-1} \text{ for } 0 \leq k \leq n. \tag{8}$$

Therefore for  $0 \leq k \leq n$ ,  $f_k(x)$  is the remainder of the division of  $x$  by  $C_k$  and it can be checked that

$$\alpha_k(x) = \left\lfloor \frac{f_k(x)}{C_{k-1}} \right\rfloor = \frac{f_k(x) - f_{k-1}(x)}{C_{k-1}} \text{ for } 1 \leq k \leq n,$$

$$\alpha_{n+1}(x) = \left\lfloor \frac{x}{C_n} \right\rfloor = \frac{x - f_n(x)}{C_n}.$$

We also define  $\Delta_k(x)$  as the integer quotient of the division of  $x$  by  $C_k$ , i.e.

$$\Delta_k(x) = \frac{x - f_k(x)}{C_k} = \sum_{j=k+1}^{n+1} \frac{C_{j-1}}{C_k} \alpha_j(x) \text{ for } 0 \leq k \leq n. \tag{9}$$

### 3 The Vertices of conv(DIV)

We consider the mixed-integer set  $DIV$  defined by (4)–(6) with the divisibility assumption. That is,  $C_0 > 0$  and for  $1 \leq k \leq n$ ,  $C_k/C_{k-1} \geq 2$  is an integer. Also  $I_j \cap I_k = \emptyset$  for  $j \neq k$  and we set  $b_l := 0$  where  $l \notin I_0 \cup \dots \cup I_n$ . For  $0 \leq k \leq n$ , define  $J_k = I_k \cup I_{k+1} \cup \dots \cup I_n \cup \{l\}$ .

We give an extended formulation for  $\text{conv}(DIV)$  with  $\mathcal{O}(mn)$  constraints and variables, where  $m = |I_0| + \dots + |I_n|$ . The first step is studying the vertices of the polyhedron  $\text{conv}(DIV)$ . Several properties of the vertices of  $\text{conv}(DIV)$  were given by Zhao and de Farias [20], who also described an algorithm to list all the vertices. We introduce here the properties that will be needed for our formulation.

Given  $s$  and an index  $1 \leq k \leq n$ , for  $t \in J_0$  define

$$b_t^k = \begin{cases} b_t + C_k & \text{if } f_k(b_t) > f_k(s) \\ b_t & \text{if } f_k(b_t) \leq f_k(s). \end{cases}$$

**Lemma 1.** *Consider indices  $0 \leq k \leq \ell$ . Then, for  $t \in I_\ell$ , the inequality*

$$\Delta_k(s) + \frac{C_\ell}{C_k} z_t \geq \Delta_k(b_t^k) \tag{10}$$

*is valid for  $\text{conv}(DIV)$  and implies inequality  $s + C_\ell z_t \geq b_t$ .*

*Proof.* Expanding  $s$  and  $b_t$  as in the first part of (9), inequality  $s + C_\ell z_t \geq b_t$  can be rewritten as

$$\Delta_k(s) + \frac{C_\ell}{C_k} z_t \geq \Delta_k(b_t) + \frac{f_k(b_t) - f_k(s)}{C_k}.$$

Since  $\ell \geq k$ ,  $\Delta_k(s) + \frac{C_\ell}{C_k} z_t$  is an integer. Therefore

$$\Delta_k(s) + \frac{C_\ell}{C_k} z_t \geq \Delta_k(b_t) + \left\lceil \frac{f_k(b_t) - f_k(s)}{C_k} \right\rceil = \Delta_k(b_t^k).$$

This also shows that (I0) implies the original inequality  $s + C_\ell z_t \geq b_t$ .  $\square$

Note that (I0) involves the term  $b_t^k$  and thus is not a linear inequality. We will show how to linearize this constraint, using the fact that for fixed  $k$ , the number  $b_t^k$  can take only two values.

**Lemma 2.** *Let  $(\bar{s}, \bar{z})$  be any vector in  $\text{conv}(DIV)$ .*

1. *Given indices  $1 \leq k \leq \ell$  and  $t \in I_\ell$ , if  $\alpha_k(\bar{s}) \neq \alpha_k(b_t^{k-1})$  then  $\bar{s} + C_\ell \bar{z}_t \geq b_t + C_{k-1}$ .*
2. *Given an index  $k \geq 1$ , if  $\alpha_k(\bar{s}) \neq 0$  then  $\bar{s} \geq C_{k-1}$ .*

*Proof.* We prove the first statement. By Lemma 1,  $(\bar{s}, \bar{z})$  satisfies (II) for the pair of indices  $k - 1, \ell$ , that is,

$$\Delta_{k-1}(s) + \frac{C_\ell}{C_{k-1}} z_t \geq \Delta_{k-1}(b_t^{k-1}).$$

By (9), the above inequality can be rewritten as

$$\sum_{j=k}^{n+1} \frac{C_{j-1}}{C_{k-1}} \alpha_j(s) + \frac{C_\ell}{C_{k-1}} z_t \geq \sum_{j=k}^{n+1} \frac{C_{j-1}}{C_{k-1}} \alpha_j(b_t^{k-1}),$$

or equivalently as

$$\sum_{j=k+1}^{n+1} \frac{C_{j-1}}{C_{k-1}} \alpha_j(s) + \frac{C_\ell}{C_{k-1}} z_t - \sum_{j=k+1}^{n+1} \frac{C_{j-1}}{C_{k-1}} \alpha_j(b_t^{k-1}) \geq \alpha_k(b_t^{k-1}) - \alpha_k(s). \quad (11)$$

Since  $\left\{ \frac{C_{j-1}}{C_{k-1}}, k < j \leq n + 1 \right\}$  is a sequence of divisible integers and since  $\ell \geq k$ , the left-hand side of the above inequality is an integer multiple of  $C_k/C_{k-1}$ . Since the right-hand side is an integer satisfying  $-C_k/C_{k-1} < \alpha_k(b_t^{k-1}) - \alpha_k(s) < C_k/C_{k-1}$ , this shows that if  $\alpha_k(\bar{s}) \neq \alpha_k(b_t^{k-1})$ , then (II) cannot be tight for  $(\bar{s}, \bar{z})$ , thus

$$\Delta_{k-1}(\bar{s}) + \frac{C_\ell}{C_{k-1}} \bar{z}_t \geq \Delta_{k-1}(b_t^{k-1}) + 1.$$

Since  $b_t^{k-1} = b_t + C_{k-1}$  if  $f_{k-1}(b_t) > f_{k-1}(\bar{s})$  and  $b_t^{k-1} = b_t$  if  $f_{k-1}(b_t) \leq f_{k-1}(\bar{s})$ , this shows that in both cases

$$\frac{f_{k-1}(\bar{s})}{C_{k-1}} + \Delta_{k-1}(\bar{s}) + \frac{C_\ell}{C_{k-1}} \bar{z}_t \geq \Delta_{k-1}(b_t) + \frac{f_{k-1}(b_t)}{C_{k-1}} + 1.$$

Multiplying the above inequality by  $C_{k-1}$  gives  $\bar{s} + C_\ell \bar{z}_t \geq b_t + C_{k-1}$ .

The proof of the second statement is an immediate consequence of expansion (7).  $\square$

**Lemma 3.** *If  $(\bar{s}, \bar{z})$  is a vertex of  $\text{conv}(DIV)$ , then the following two properties hold:*

1.  $\alpha_0(\bar{s}) = \alpha_0(b_t)$  for some  $t \in J_0$ .
2. For  $1 \leq k \leq n$ ,  $\alpha_k(\bar{s}) = \alpha_k(b_t^{k-1})$  for some  $t \in J_k$ .

*Proof.* Let  $(\bar{s}, \bar{z})$  be a vertex of  $\text{conv}(DIV)$ . Since  $\bar{z}$  is an integral vector, if 1. is violated then there is  $\varepsilon \neq 0$  such that  $(\bar{s} \pm \varepsilon, \bar{z}) \in \text{conv}(DIV)$ , a contradiction.

Assume that 2. is violated, i.e. there is an index  $k$  such that  $\alpha_k(\bar{s}) \neq \alpha_k(b_t^{k-1})$  for all  $t \in J_k$ . In particular, for  $t = l$  we have  $\alpha_k(\bar{s}) \neq 0$ . Consider the vector  $v_{k-1}$  defined as follows:

$$s = -C_{k-1}, \quad z_t = \frac{C_{k-1}}{C_\ell}, \quad t \in I_\ell, \ell \leq k-1, \quad z_t = 0, \quad t \in I_\ell, \ell > k-1.$$

By Lemma 2 we have that  $s \geq C_{k-1}$  and  $\bar{s} + C_\ell \bar{z}_t \geq b_t + C_{k-1}$  for  $t \in I_\ell, \ell \geq k$ . This shows that the vectors  $(\bar{s}, \bar{z}) \pm v_{k-1}$  belong to  $\text{conv}(DIV)$ . Hence  $(\bar{s}, \bar{z})$  is not a vertex of  $\text{conv}(DIV)$ . □

We now introduce extra variables to model the possible values taken by  $s$  at a vertex of  $\text{conv}(DIV)$ . The new variables are the following:

- $\Delta_0, w_{0,t}$  for  $t \in J_0$ ;
- $\Delta_k, w_{k,t}^\downarrow, w_{k,t}^\uparrow$  for  $1 \leq k \leq n$  and  $t \in J_k$ .

The role of the above variables is as follows:

- Variables  $\Delta_k$  are the integer quotients of the division of  $s$  by  $C_k$ . That is,  $\Delta_k = \Delta_k(s)$  as defined in (9).
- Variable  $w_{0,t} = 1$  whenever  $\alpha_0(s) = \alpha_0(b_t)$  and  $w_{0,t} = 0$  otherwise.
- Variable  $w_{k,t}^\downarrow = 1$  whenever  $\alpha_k(s) = \alpha_k(b_t)$  and  $w_{k,t}^\uparrow = 1$  whenever  $\alpha_k(s) = \alpha_k(b_t + C_{k-1})$ ;  $w_{k,t}^\downarrow = w_{k,t}^\uparrow = 0$  otherwise.

Consider the following conditions:

$$s = C_0 \Delta_0 + \sum_{i \in J_0} \alpha_0(b_i) w_{0,i}, \tag{12}$$

$$\Delta_{k-1} = \frac{C_k}{C_{k-1}} \Delta_k + \sum_{i \in J_k} \left( \alpha_k(b_i) w_{k,i}^\downarrow + \alpha_k(b_i + C_{k-1}) w_{k,i}^\uparrow \right), \quad 1 \leq k \leq n, \tag{13}$$

$$w_{0,i} \geq 0, \quad i \in J_0; \quad \sum_{i \in J_0} w_{0,i} = 1, \tag{14}$$

$$w_{k,i}^\downarrow, w_{k,i}^\uparrow \geq 0, \quad i \in J_k, \quad 1 \leq k \leq n; \quad \sum_{i \in J_k} \left( w_{k,i}^\downarrow + w_{k,i}^\uparrow \right) = 1, \quad 1 \leq k \leq n, \tag{15}$$

$$\sum_{\substack{i \in J_0: \\ \alpha_0(b_i) \geq \alpha_0(b_t)}} w_{0,i} \geq w_{1,t}^\downarrow, \quad t \in J_1, \tag{16}$$

$$\sum_{\substack{i \in J_k: \\ f_k(b_i) \geq f_k(b_t)}} w_{k,i}^\downarrow + \sum_{\substack{i \in J_k: \\ \alpha_k(b_i + C_{k-1}) \geq \alpha_k(b_t) + 1}} w_{k,i}^\uparrow \geq w_{k+1,t}^\downarrow, \quad t \in J_{k+1}, 1 \leq k \leq n-1, \tag{17}$$

$$\Delta_k, w_{0,i}, w_{k,i}^\downarrow, w_{k,i}^\uparrow \text{ integer, } i \in J_k, 0 \leq k \leq n. \tag{18}$$

**Lemma 4.** *If  $(\bar{s}, \bar{z})$  is a vertex of  $\text{conv}(DIV)$ , then  $(\bar{s}, \bar{z})$  can be completed to a vector  $(\bar{s}, \bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\downarrow, \bar{w}^\uparrow)$  satisfying (12)–(18).*

*Proof.* Given vertex  $(\bar{s}, \bar{z})$ , let  $i_0$  be any index in  $J_0$  such that  $\alpha_0(b_{i_0}) = \alpha_0(\bar{s})$  ( $i_0$  exists by Lemma 3). Take  $\bar{w}_{0,i_0} = 1$  and  $\bar{w}_{0,i} = 0$  for  $i \neq i_0$ . Now fix  $k \geq 1$  and define

$$T_k(\bar{s}) = \{i \in J_k : \alpha_k(\bar{s}) = \alpha_k(b_i), f_{k-1}(\bar{s}) \geq f_{k-1}(b_i)\}.$$

If  $T_k(\bar{s}) \neq \emptyset$ , then define  $i_k$  as any element in  $T_k(\bar{s})$  such that  $f_{k-1}(b_{i_k})$  is maximum and take  $\bar{w}_{k,i_k}^\downarrow = 1$ . Otherwise ( $T_k(\bar{s}) = \emptyset$ ) define  $i_k$  as any index in  $J_k$  such that  $\alpha_k(\bar{s}) = \alpha_k(b_{i_k} + C_{k-1})$  ( $i_k$  exists by Lemma 3) and take  $\bar{w}_{k,i_k}^\uparrow = 1$ . Finally take  $\bar{\Delta}_k = \Delta_k(\bar{s})$  for  $0 \leq k \leq n$ .

We prove that the point thus constructed satisfies (12)–(18). To see that (12) is satisfied, note that

$$C_0 \bar{\Delta}_0 + \sum_{i \in J_0} \alpha_0(b_i) \bar{w}_{0,i} = C_0 \Delta_0(\bar{s}) + \alpha_0(b_{i_0}) = C_0 \Delta_0(\bar{s}) + f_0(b_{i_0}) = \bar{s}.$$

To prove (13), note that the following chain of equations holds:

$$\begin{aligned} \frac{C_k}{C_{k-1}} \bar{\Delta}_k + \sum_{i \in J_k} \left( \alpha_k(b_i) \bar{w}_{k,i}^\downarrow + \alpha_k(b_i + C_{k-1}) \bar{w}_{k,i}^\uparrow \right) \\ = \frac{C_k}{C_{k-1}} \Delta_k(\bar{s}) + \alpha_k(\bar{s}) = \Delta_{k-1}(\bar{s}) = \bar{\Delta}_{k-1}. \end{aligned}$$

To see that (16) is verified, suppose that  $\bar{w}_{1,t}^\downarrow = 1$  for an index  $t \in J_1$ . Then necessarily  $t = i_1 \in T_1(\bar{s})$  and thus  $f_0(\bar{s}) \geq f_0(b_t)$ , that is,  $\alpha_0(\bar{s}) \geq \alpha_0(b_t)$ . Then  $\alpha_0(b_{i_0}) = \alpha_0(\bar{s}) \geq \alpha_0(b_t)$  and (16) is satisfied.

We now consider (17) for  $k \geq 1$ . Suppose that  $\bar{w}_{k+1,t}^\downarrow = 1$  for an index  $t \in J_{k+1}$ . Then necessarily  $t = i_{k+1} \in T_{k+1}(\bar{s})$ . Therefore  $\alpha_{k+1}(\bar{s}) = \alpha_{k+1}(b_t)$  and  $f_k(\bar{s}) \geq f_k(b_t)$ . This implies  $\alpha_k(\bar{s}) \geq \alpha_k(b_t)$ . We distinguish two cases.

1. Assume  $\alpha_k(\bar{s}) \geq \alpha_k(b_t) + 1$ . If  $T_k(\bar{s}) \neq \emptyset$  then  $\bar{w}_{k,i}^\downarrow = 1$  for an index  $i \in J_k$  such that  $\alpha_k(b_i) = \alpha_k(\bar{s}) \geq \alpha_k(b_t) + 1$ . Then  $f_k(b_i) \geq f_k(b_t)$ . If  $T_k(\bar{s}) = \emptyset$  then  $\bar{w}_{k,i}^\uparrow = 1$  for an index  $i \in J_k$  such that  $\alpha_k(b_i + C_{k-1}) = \alpha_k(\bar{s}) \geq \alpha_k(b_t) + 1$ . In both cases (17) is satisfied.

2. Now assume  $\alpha_k(\bar{s}) = \alpha_k(b_t)$ . In this case inequality  $f_k(\bar{s}) \geq f_k(b_t)$  implies  $f_{k-1}(\bar{s}) \geq f_{k-1}(b_t)$ , thus  $t \in T_k(\bar{s}) \neq \emptyset$ . Then the choice of  $i_k$  shows that  $\alpha_k(b_{i_k}) = \alpha_k(\bar{s}) = \alpha_k(b_t)$  and  $f_{k-1}(b_{i_k}) \geq f_{k-1}(b_t)$ , thus  $f_k(b_{i_k}) \geq f_k(b_t)$  and (17) is satisfied.

Constraints (14)–(15) and (18) are clearly satisfied. □

We say that  $(\bar{s}, \bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\downarrow, \bar{w}^\uparrow)$  is a *standard completion* of a vertex  $(\bar{s}, \bar{z})$  if  $\bar{\Delta}, \bar{w}, \bar{w}^\downarrow, \bar{w}^\uparrow$  are chosen as in the above proof. Then the above proof shows that every vertex of  $\text{conv}(DIV)$  has a standard completion satisfying (12)–(18).

**Lemma 5.** *If  $(\bar{s}, \bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\downarrow, \bar{w}^\uparrow)$  satisfies (12)–(18), then*

$$f_0(\bar{s}) \geq f_0(b_t) \text{ if } \sum_{\substack{i \in J_0: \\ \alpha_0(b_i) \geq \alpha_0(b_t)}} w_{0,i} = 1, \quad t \in J_0,$$

$$f_k(\bar{s}) \geq f_k(b_t) \text{ if } \sum_{\substack{i \in J_k: \\ f_k(b_i) \geq f_k(b_t)}} w_{k,i}^\downarrow + \sum_{\substack{i \in J_k: \\ \alpha_k(b_i + C_{k-1}) \geq \alpha_k(b_t) + 1}} w_{k,i}^\uparrow = 1, \quad t \in J_k, k \geq 1.$$

*Proof.* Let  $t \in J_0$  and assume that

$$\sum_{\substack{i \in J_0: \\ \alpha_0(b_i) \geq \alpha_0(b_t)}} \bar{w}_{0,i} = 1$$

holds. If  $i \in J_0$  is the index such that  $\bar{w}_{0,i} = 1$  then, by (12),  $f_0(\bar{s}) = \alpha_0(b_i) \geq \alpha_0(b_t) = f_0(b_t)$ .

We now fix  $0 \leq k < n$  and assume by induction that the result holds for any index  $t \in J_k$ . We have to prove that if

$$\sum_{\substack{i \in J_{k+1}: \\ f_{k+1}(b_i) \geq f_{k+1}(b_t)}} w_{k+1,i}^\downarrow + \sum_{\substack{i \in J_{k+1}: \\ \alpha_{k+1}(b_i + C_k) \geq \alpha_{k+1}(b_t) + 1}} w_{k+1,i}^\uparrow = 1 \tag{19}$$

for some  $t \in J_{k+1}$ , then  $f_{k+1}(\bar{s}) \geq f_{k+1}(b_t)$ .

If  $\bar{w}_{k+1,i}^\uparrow = 1$  for some index  $i \in J_{k+1}$ , then (13) and the above equation give  $\alpha_{k+1}(\bar{s}) = \alpha_{k+1}(b_i + C_k) \geq \alpha_{k+1}(b_t) + 1$ , thus  $f_{k+1}(\bar{s}) \geq f_{k+1}(b_t)$ .

If  $\bar{w}_{k+1,i}^\downarrow = 1$  for some index  $i \in J_{k+1}$ , then (19) implies that  $f_{k+1}(b_i) \geq f_{k+1}(b_t)$ , thus  $\alpha_{k+1}(b_i) \geq \alpha_{k+1}(b_t)$ . Assume first that  $\alpha_{k+1}(b_i) \geq \alpha_{k+1}(b_t) + 1$ . Then  $\alpha_{k+1}(\bar{s}) = \alpha_{k+1}(b_i) \geq \alpha_{k+1}(b_t) + 1$ , thus  $f_{k+1}(\bar{s}) \geq f_{k+1}(b_t)$ .

Finally assume that  $\bar{w}_{k+1,i}^\downarrow = 1$  for some  $i \in J_{k+1}$  such that  $\alpha_{k+1}(b_i) = \alpha_{k+1}(b_t)$ . Since (19) implies  $f_{k+1}(b_i) \geq f_{k+1}(b_t)$ , we then have  $f_k(b_i) \geq f_k(b_t)$ . Inequality (17) for the index  $i$  implies that

$$\sum_{\substack{j \in J_k: \\ f_k(b_j) \geq f_k(b_i)}} \bar{w}_{k,j}^\downarrow + \sum_{\substack{j \in J_k: \\ \alpha_k(b_j + C_{k-1}) \geq \alpha_k(b_i) + 1}} \bar{w}_{k,j}^\uparrow = 1.$$

Then, by induction,  $f_k(\bar{s}) \geq f_k(b_i)$ . This, together with inequality  $f_k(b_i) \geq f_k(b_t)$  proven above, shows that  $f_k(\bar{s}) \geq f_k(b_t)$ . Using  $\alpha_{k+1}(\bar{s}) = \alpha_{k+1}(b_i) = \alpha_{k+1}(b_t)$ , we conclude that  $f_{k+1}(\bar{s}) \geq f_{k+1}(b_t)$ . □



Lemma 5 and the same argument used in the final part of the proof of Lemma 4 prove the following:

Remark 6. If  $(\bar{s}, \bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\downarrow, \bar{w}^\uparrow)$  is a standard completion of a vertex  $(\bar{s}, \bar{z})$  of  $\text{conv}(DIV)$ , then

$$f_0(s) \geq f_0(b_t) \iff \sum_{\substack{i \in J_0: \\ \alpha_0(b_i) \geq \alpha_0(b_t)}} w_{0,i} = 1, \quad t \in J_0,$$

$$f_k(s) \geq f_k(b_t) \iff \sum_{\substack{i \in J_k: \\ f_k(b_i) \geq f_k(b_t)}} w_{k,i}^\downarrow + \sum_{\substack{i \in J_k: \\ \alpha_k(b_i + C_{k-1}) \geq \alpha_k(b_t) + 1}} w_{k,i}^\uparrow = 1, \quad t \in J_k, k \geq 1.$$

### 4 Linearizing (10)

Lemma 7. Let  $(s, z, \Delta, w, w^\uparrow, w^\downarrow)$  be a vector satisfying (12)–(18). Then  $(s, z)$  satisfies inequality  $s + C_k z_t \geq b_t$  if and only if  $(s, z, \Delta, w, w^\uparrow, w^\downarrow)$  satisfies the inequality:

$$\Delta_0 + \sum_{\substack{i \in J_0: \\ \alpha_0(b_i) \geq \alpha_0(b_t)}} w_{0,i} + z_t \geq \left\lfloor \frac{b_t}{C_0} \right\rfloor + 1 \quad \text{if } t \in J_0, \tag{20}$$

$$\Delta_k + \sum_{\substack{i \in J_k: \\ f_k(b_i) \geq f_k(b_t)}} w_{k,i}^\downarrow + \sum_{\substack{i \in J_k: \\ \alpha_k(b_i + C_{k-1}) \geq \alpha_k(b_t) + 1}} w_{k,i}^\uparrow + z_t \geq \left\lfloor \frac{b_t}{C_k} \right\rfloor + 1$$

if  $t \in J_k, k \geq 1. \tag{21}$

Proof. We prove the following two facts: (i) if  $(s, z, \Delta, w, w^\uparrow, w^\downarrow)$  is a standard completion of a vertex of  $\text{conv}(DIV)$ , then (20)–(21) hold; (ii) if the vector  $(s, z, \Delta, w, w^\uparrow, w^\downarrow)$  satisfies (12)–(18) along with (20) (if  $t \in J_0$ ) or (21) (if  $t \in J_k$  with  $k \geq 1$ ), then it also satisfies  $s + C_k z_t \geq b_t$ .

By Lemma 1, inequality  $s + C_\ell z_t \geq b_t$  is equivalent to  $\Delta_k(s) + \frac{C_\ell}{C_k} z_t \geq \Delta_k(b_t^k)$  for  $\ell \geq k$ . In particular, for  $\ell = k$  the latter inequality is in turn equivalent to the inequality  $\Delta_k(s) + z_t + \delta \geq \Delta_k(b_t + C_k) = \left\lfloor \frac{b_t}{C_k} \right\rfloor + 1$ , where  $\delta$  is a 0, 1 variable that takes value 1 whenever  $f_k(s) \geq f_k(b_t)$  and 0 otherwise.

If  $t \in J_0$ , by Remark 6 a standard completion  $(\bar{s}, \bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\uparrow, \bar{w}^\downarrow)$  of any vertex  $(\bar{s}, \bar{z})$  of  $\text{conv}(DIV)$  satisfies

$$\sum_{\substack{i \in J_0: \\ \alpha_0(b_i) \geq \alpha_0(b_t)}} w_{0,i} = 1 \iff f_0(s) \geq f_0(b_t).$$

Then substituting the above expression for  $\delta$  shows that  $(\bar{s}, \bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\uparrow, \bar{w}^\downarrow)$  satisfies (20). If  $t \in J_k$  with  $k \geq 1$ , the proof that  $(\bar{s}, \bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\uparrow, \bar{w}^\downarrow)$  satisfies (21) is similar. This proves (i).

By Lemma 5 with the above definition of  $\delta$ , one observes that  $\delta = 0$  for every vector  $(s, z, \Delta, w, w^\uparrow, w^\downarrow)$  satisfying (12)–(18) such that  $f_k(s) < f_k(b_t)$ . This implies (ii).  $\square$

The following result is readily checked:

*Remark 8.* Let  $(s, z, \Delta, w, w^\uparrow, w^\downarrow)$  be a vector satisfying (12)–(18). Then  $(s, z)$  satisfies inequality  $s \geq 0$  if and only if  $(s, z, \Delta, w, w^\uparrow, w^\downarrow)$  satisfies the inequality

$$\Delta_n \geq 0. \tag{22}$$

### 5 Strengthening (16)–(17)

**Lemma 9.** *The following inequalities are valid for the set defined by (12)–(18) and dominate (16)–(17):*

$$\begin{aligned} \sum_{\substack{i \in J_0: \\ \alpha_0(b_i) \geq \alpha_0(b_t)}} w_{0,i} &\geq \sum_{\substack{i \in J_1: \\ f_0(b_i) \geq f_0(b_t)}} w_{1,i}^\downarrow, \quad t \in J_1, \tag{23} \\ \sum_{\substack{i \in J_k: \\ f_k(b_i) \geq f_k(b_t)}} w_{k,i}^\downarrow + \sum_{\substack{i \in J_k: \\ \alpha_k(b_i + C_{k-1}) \geq \alpha_k(b_t) + 1}} w_{k,i}^\uparrow &\geq \sum_{\substack{i \in J_{k+1}: \\ f_k(b_i) \geq f_k(b_t)}} w_{k+1,i}^\downarrow, \\ &t \in J_{k+1}, \quad 1 \leq k \leq n-1. \tag{24} \end{aligned}$$

*Proof.* Fix  $t \in J_{k+1}$  for  $k \geq 1$  and define  $L = \{i \in J_{k+1} : f_k(b_i) \geq f_k(b_t)\}$ . Inequality (24) can be derived by applying the Chvátal-Gomory procedure to the following  $|L| + 1$  inequalities, which are all valid for (12)–(18):

$$\sum_{\substack{i \in J_k: \\ f_k(b_i) \geq f_k(b_j)}} w_{k,i}^\downarrow + \sum_{\substack{i \in J_k: \\ \alpha_k(b_i + C_{k-1}) \geq \alpha_k(b_j) + 1}} w_{k,i}^\uparrow \geq w_{k+1,j}^\downarrow, \quad j \in L, \tag{25}$$

$$1 \geq \sum_{j \in L} w_{k+1,j}^\downarrow, \tag{26}$$

with multipliers  $1/|L|$  for each of (25) and  $1 - 1/|L|$  for (26). The derivation of (23) is similar.  $\square$

### 6 The Main Result

Let  $Q$  be the polyhedron in the space of variables  $x = (s, z, \Delta, w, w^\downarrow, w^\uparrow)$  defined by (12)–(15) together with (20)–(21), (22) and (23)–(24). We denote by  $Ax \sim b$  the system comprising such equations and inequalities.

**Lemma 10.** *Let  $M$  be the submatrix of  $A$  indexed by the columns corresponding to variables  $w, w^\downarrow, w^\uparrow$  and the rows corresponding to (14)–(15) and (23)–(24). The matrix  $M$  is totally unimodular.*

*Proof.* We use a characterization of Ghouila-Houri [8], which states that a  $0, \pm 1$  matrix  $B = (b_{ij})$  is totally unimodular if and only if for every row submatrix  $B'$  of  $B$ , the set of row indices of  $B'$  can be partitioned into two subsets  $R_1, R_2$  such that  $\sum_{i \in R_1} b_{ij} - \sum_{i \in R_2} b_{ij} \in \{0, \pm 1\}$  for all column indices  $j$ .

We partition the rows of  $M$  into the submatrices  $M_0, \dots, M_n$  defined as follows:

- $M_0$  consists of the rows corresponding to equation (14) and inequalities (23) for  $t \in J_1$ ;
- for  $1 \leq k \leq n - 1$ ,  $M_k$  consists of the rows corresponding to equation (15) and inequalities (24) for  $t \in J_{k+1}$ ;
- $M_n$  consists of the row corresponding to equation (15) for  $k = n$ .

For each odd  $k$ , we multiply by  $-1$  the rows of  $M$  that belongs to  $M_k$  and the columns of  $M$  corresponding to variables  $w_{k,t}^\downarrow, w_{k,t}^\uparrow$  for all  $t \in J_k$ . Then  $M$  becomes a 0-1 matrix.

For  $1 \leq k \leq n - 1$ , we order the rows of  $M_k$  as follows: first the row corresponding to (15), then those corresponding to (24) according to a non-decreasing order of the values  $f_k(b_t)$ . The order for the rows of  $M_0$  is analogous. Note that in every matrix  $M_k$  the support of any row, say the  $j$ -th row, contains that of the  $(j + 1)$ -th row (in other words, the rows of  $M_k$  form a laminar family).

We now define a bipartition  $(R_1, R_2)$  of the rows of  $M$ : for each odd  $k$ , we include in  $R_1$  the odd row indices of  $M_k$  and in  $R_2$  the even row indices; for each even  $k$ , we include in  $R_1$  the even row indices of  $M_k$  and in  $R_2$  the odd row indices. One can check that the condition of the theorem of Ghouila-Houri is thus satisfied for  $B' = M$ . If  $B'$  is a row submatrix of  $M$ , the bipartition is defined similarly. □

**Theorem 11.** *If  $\bar{x} = (\bar{s}, \bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\downarrow, \bar{w}^\uparrow)$  is a vertex of  $Q$ , then  $(\bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\downarrow, \bar{w}^\uparrow)$  is an integral vector. It follows that the inequalities defining  $Q$  provide an extended formulation for the polyhedron  $\text{conv}(DIV)$  with  $\mathcal{O}(mn)$  variables and constraints, where  $m = |I_0| + \dots + |I_n|$ .*

*Proof.* Note that the columns of  $A$  corresponding to variables  $s$  and  $z_t$  for  $t \in I_k$  and  $0 \leq k \leq n$  are unit columns (as  $s$  only appears in (12) and each variable  $z_t$  only appears in one of (20)–(21)).

Also note that in the subsystem of  $Ax \sim b$  comprising (13)–(15), (22) and (23)–(24) (i.e. with (12) and (20)–(21) removed) variables  $\Delta_0, \dots, \Delta_n$  appear with nonzero coefficient only in (13) and (22). Furthermore the submatrix of  $A$  indexed by the rows corresponding to (13) and (22) and the columns corresponding to variables  $\Delta_0, \dots, \Delta_n$  is an upper triangular matrix with 1 on the diagonal.

Let  $Cx = d$  be a nonsingular subsystem of tight inequalities taken in  $Ax \sim b$  that defines a vertex  $\bar{x} = (\bar{s}, \bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\downarrow, \bar{w}^\uparrow)$  of  $Q$ . The above observations show that (12)–(13), (20)–(21) and (22) must be present in this subsystem. Furthermore let  $C'$  be the submatrix of  $C$  indexed by the columns corresponding to variables  $w, w^\downarrow, w^\uparrow$  and the rows that do not correspond to (12)–(13),

(20)–(21) and (22). Then the computation of a determinant with Laplace expansion shows that  $|\det(C)| = |\det(C')| \neq 0$ .

Since  $C'$  is a nonsingular submatrix of the matrix  $M$  defined in Lemma 10, by Lemma 10  $|\det(C)| = |\det(C')| = 1$ . Since all entries of  $A$  (except those corresponding to (12)) are integer and the right-hand side vector  $b$  is integral, by Cramer’s rule we have that  $(\bar{z}, \bar{\Delta}, \bar{w}, \bar{w}^\perp, \bar{w}^\perp)$  is an integral vector.  $\square$

## 7 The Mixing Set with Divisible Capacities and Nonnegative Integer Variables

The mixing set with divisible capacities and nonnegativity bounds on the integer variables  $DIV^+$  is the following:

$$\begin{aligned} s + C_k z_t &\geq b_t, & t \in I_k, 0 \leq k \leq n, \\ b_l &\leq s \leq b_u, \\ z_t &\geq 0 \text{ integer}, & t \in I_0 \cup \dots \cup I_n, \end{aligned}$$

where the capacities  $C_k$ ’s and the sets  $I_k$ ’s are as in the previous sections.

Di Summa [6] gave a polynomial time algorithm to optimize a linear function over  $DIV^+$ . We discuss the problem of finding an extended formulation for the polyhedron  $\text{conv}(DIV^+)$  which is compact.

We do not know how to incorporate the bounds  $z_t \geq 0$  in a formulation of the type given for the polyhedron  $Q$  of Theorem 11, as the standard approach requires that the system, purged of the equations defining  $s$  and  $\Delta_k$ , be defined by a totally unimodular matrix (see for instance [3,12,15,18,19]). However this is not the case, as discussed in the next paragraph. So we use an approach based on union of polyhedra in a manner described e.g. in [14].

To this purpose, let  $\{\beta_1, \dots, \beta_q\}$  be the set of distinct values in the set  $\{b_i : i \in I_0 \cup \dots \cup I_n, b_l < b_i < b_u\}$ . Assume  $\beta_1 < \dots < \beta_q$  and define  $\beta_0 := b_l$  and  $\beta_{q+1} := b_u$ . For each  $0 \leq \ell \leq q$ , let  $DIV(\ell)$  be the following set:

$$\begin{aligned} s + C_k z_i &\geq b_i, & i \in I_k : b_i > \beta_\ell, 0 \leq k \leq m, \\ \beta_\ell &\leq s \leq \beta_{\ell+1}, \\ z_i &\geq 0, & i \in I_k : b_i \leq \beta_\ell, 0 \leq k \leq m, \\ z_i &\text{ integer}, & i \in I_0 \cup \dots \cup I_m. \end{aligned}$$

We will use the following fact:

$$\text{conv}(DIV^+) = \text{conv} \left( \bigcup_{\ell=1}^q DIV(\ell) \right). \tag{27}$$

We now examine the problem of finding extended formulations which are compact for the polyhedra  $\text{conv}(DIV(\ell))$ . Note that  $DIV(\ell)$  is the cartesian product of the following two sets:

$$\begin{aligned}
 s + C_k z_i &\geq b_i, & i \in I_k : b_i > \beta_\ell, 0 \leq k \leq m, \\
 \beta_\ell &\leq s \leq \beta_{\ell+1}, \\
 z_i &\text{ integer}, & i \in I_k : b_i > \beta_\ell, 0 \leq k \leq m,
 \end{aligned}$$

and

$$\begin{aligned}
 z_i &\geq 0, & i \in I_k : b_i \leq \beta_\ell, 0 \leq k \leq m, \\
 z_i &\text{ integer}, & i \in I_k : b_i \leq \beta_\ell, 0 \leq k \leq m.
 \end{aligned}$$

If we denote by  $UDIV(\ell)$  the first of the above two sets, then  $\text{conv}(DIV(\ell)) = \text{conv}(UDIV(\ell)) \times \{z : z_i \geq 0\}$ .

Remark that  $UDIV(\ell)$  is a mixing set with divisible capacities without non-negativity bounds on the integer variables, except that now we have an upper bound  $s \leq \beta_{\ell+1}$ . A compact extended formulation for  $UDIV(\ell)$  can be derived by using the same ideas presented in this paper (but there are more technicalities) and can be found in [7].

Using (27) and a classical result of Balas [2], a compact extended formulation for  $\text{conv}(DIV^+)$  can be derived from the compact extended formulations of the  $q$  polyhedra  $\text{conv}(DIV(\ell))$ .

### 7.1 An Instance with Non-TU Matrix

We show an instance of  $DIV$  for which the formulation given by the inequalities describing  $Q$  in Theorem 11, purged of the equations defining  $s$  and  $\Delta_k$ , is not defined by a totally unimodular matrix. The instance is the following:

$$\begin{aligned}
 s + z_1 &\geq 0.1, \\
 s + 10z_2 &\geq 6.3, \\
 s + 100z_3 &\geq 81.4, \\
 s + 100z_4 &\geq 48.6, \\
 s &\geq 0; z_1, \dots, z_4 \text{ integer.}
 \end{aligned}$$

Note that  $I_0 = \{1\}$ ,  $I_1 = \{2\}$  and  $I_3 = \{3, 4\}$ .

Among the constraints defining the extended formulation of the convex hull of the above set, we consider the following four inequalities:

$$\begin{aligned}
 w_{1,2}^\downarrow + w_{1,2}^\uparrow + w_{1,3}^\downarrow + w_{1,3}^\uparrow + w_{1,4}^\downarrow + w_{1,4}^\uparrow &\geq w_{2,3}^\downarrow + w_{2,4}^\downarrow, \\
 w_{0,3} + w_{0,4} &\geq w_{1,3}^\downarrow + w_{1,4}^\downarrow, \\
 w_{1,4}^\downarrow + w_{1,4}^\uparrow &\geq w_{2,4}^\downarrow, \\
 \Delta_1 + w_{1,2}^\downarrow + w_{1,2}^\uparrow + w_{1,4}^\downarrow + w_{1,4}^\uparrow + z_2 &\geq 1,
 \end{aligned}$$

which correspond respectively to (24) for  $k = 1$  and  $t = 3$ , (23) for  $t = 3$ , (24) for  $k = 1$  and  $t = 4$ , and (21) for  $k = 1$  and  $t = 2$ .

The constraint matrix of the above four inequalities is not totally unimodular, as the determinant of the column submatrix corresponding to variables  $w_{1,4}^\downarrow, w_{1,3}^\downarrow, w_{2,4}^\downarrow, w_{1,2}^\uparrow$  is  $-2$ .

## 8 Remarks and Open Questions

- The extended formulation presented here is based on the expansion  $x = \alpha_0(x) + \sum_{j=1}^{n+1} \alpha_j(x)C_{j-1}$  of a real number  $x$  and then exploits the fact that, if  $\bar{x}$  is a vertex of the polyhedron to be studied, then for fixed  $0 \leq j \leq n+1$ , there are few values that  $\alpha_j(\bar{x})$  can take. This is essential for the extended formulation to be compact.

This can be seen as a nontrivial extension of the technique used by Miller and Wolsey [12] in the single capacity mixing set (i.e.  $n = 0$ ) to model a continuous variable  $x$  by taking  $C_0 = 1$  and  $x = \alpha_0(x) + \alpha_1(x)C_0$ . Indeed, if one imposes in *DIV* the further restriction that  $s$  is integer (which removes all the complexity in the single capacity mixing set), the complexity of *DIV* remains essentially unchanged.

- *CAP* is the following mixed-integer set:

$$\begin{aligned} s_i + C_t z_t &\geq b_{it}, & 1 \leq i \leq q, 0 \leq t \leq m, \\ s_i &\geq b_{li}, & 1 \leq i \leq q, \\ z_t &\text{integer}, & 0 \leq t \leq m, \end{aligned}$$

where again  $C_0, \dots, C_m$  is a sequence of divisible numbers. Note that the set *DIV* is a special case of *CAP*, obtained by taking  $q = 1$ . What is the complexity of optimizing a linear function over *CAP*? Does *CAP* admit a formulation that is computationally useful? These questions were investigated and answered by Miller and Wolsey [12] for the single capacity case.

- Our last question concerns the mixing set with arbitrary capacities, defined by (1)–(3) in the introduction of this paper. Again, what is the complexity of optimizing a linear function over (1)–(3)? In the case where the number of distinct capacities is small, does there exist an extended formulation which is compact?

## References

1. Atamtürk, A.: Strong formulations of robust mixed 0-1 programming. *Mathematical Programming* 108, 235–250 (2006)
2. Balas, E.: Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics* 89, 3–44 (1998)
3. Conforti, M., Di Summa, M., Eisenbrand, F., Wolsey, L.A.: Network formulations of mixed-integer programs. CORE Discussion Paper, 2006/117, Université catholique de Louvain, Belgium. *Mathematics of Operations Research* (accepted, 2006)
4. Conforti, M., Wolsey, L.A.: Compact formulations as a union of polyhedra. *Mathematical Programming* (published online) (to appear, 2007)
5. Constantino, M., Miller, A.J., Van Vyve, M.: Mixing MIR inequalities with two divisible coefficients (manuscript, 2007)
6. Di Summa, M.: The mixing set with divisible capacities (manuscript, 2007)
7. Di Summa, M.: Formulations of Mixed-Integer Sets Defined by Totally Unimodular Constraint Matrices. PhD thesis, Università degli Studi di Padova, Italy (2008)

8. Ghouila-Houri, A.: Caractérisations des matrices totalement unimodulaires. In: *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences, Paris*, vol. 254, pp. 1192–1194 (1962)
9. Günlük, O., Pochet, Y.: Mixing mixed-integer inequalities. *Mathematical Programming* 90, 429–457 (2001)
10. Henk, M., Weismantel, R.: Diophantine approximations and integer points of cones. *Combinatorica* 22, 401–408 (2002)
11. Marcotte, O.: The cutting stock problem and integer rounding. *Mathematical Programming* 33, 82–92 (1985)
12. Miller, A.J., Wolsey, L.A.: Tight formulations for some simple mixed integer programs and convex objective integer programs. *Mathematical Programming* 98, 73–88 (2003)
13. Pochet, Y., Weismantel, R.: The sequential knapsack polytope. *SIAM Journal on Optimization* 8, 248–264 (1998)
14. Pochet, Y., Wolsey, L.A.: Network design with divisible capacities: Aggregated flow and knapsack subproblems. In: Balas, E., Cornuéjols, G., Kannan, R. (eds.) *Integer Programming and Combinatorial Optimization*, pp. 324–336. Carnegie Mellon University (1992)
15. Pochet, Y., Wolsey, L.A.: Polyhedra for lot-sizing with Wagner-Whitin costs. *Mathematical Programming* 67, 297–323 (1994)
16. Pochet, Y., Wolsey, L.A.: Integer knapsack and flow covers with divisible coefficients: Polyhedra, optimization and separation. *Discrete Applied Mathematics* 59, 57–74 (1995)
17. Pochet, Y., Wolsey, L.A.: *Production Planning by Mixed Integer Programming*. Springer, Heidelberg (2006)
18. Van Vyve, M.: A Solution Approach of Production Planning Problems Based on Compact Formulations for Single-Item Lot-Sizing Models. PhD thesis, Faculté des Sciences Appliquées, Université catholique de Louvain, Belgium (2003)
19. Van Vyve, M.: Linear-programming extended formulations for the single-item lot-sizing problem with backloging and constant capacity. *Mathematical Programming* 108, 53–77 (2006)
20. Zhao, M., de Farias Jr., I.R.: The mixing-MIR set with divisible capacities. *Mathematical Programming* (published online) (to appear, 2007)

# A Polynomial Time Algorithm for the Stochastic Uncapacitated Lot-Sizing Problem with Backlogging

Yongpei Guan<sup>1,\*</sup> and Andrew Miller<sup>2</sup>

<sup>1</sup> School of Industrial Engineering, University of Oklahoma, USA  
yguan@ou.edu

<sup>2</sup> Department of Industrial and Systems Engineering, University of Wisconsin, USA  
amiller@engr.wisc.edu

**Abstract.** Since Wagner and Whitin published a seminal paper on the deterministic uncapacitated lot-sizing problem, many other researchers have investigated the structure of other fundamental models on lot-sizing that are embedded in practical production planning problems. In this paper we consider basic versions of such models in which demand (and other problem parameters) are stochastic rather than deterministic. It is named stochastic uncapacitated lot-sizing problem with backlogging. We define a production path property of optimal solutions for this model and use this property to develop backward dynamic programming recursions. This approach allows us to show that the value function is piecewise linear and continuous, which we can further use to define a polynomial time algorithm for the problem in a general stochastic scenario tree setting.

## 1 Introduction

The deterministic uncapacitated lot-sizing problem (ULS) is the problem of determining the amount to produce in each time period over a finite discrete horizon so as to satisfy the demand for each period while minimizing the summation of setup, production, and inventory holding costs (e.g., see [16], [25], and [17]). This fundamental model is easily seen to be a fixed charge network flow problem in a network composed of a directed path and a dummy production node, and understanding and exploiting this structure has been essential in developing approaches to more complicated, real-world problems (see, for example, [22], [4], [5], [20], and many others).

Polynomial time algorithms for ULS are based on the Wagner-Whitin property: no production is undertaken if inventory is available from the previous time period. A simple dynamic programming algorithm based on this property runs in  $\mathcal{O}(T^2)$  time, where  $T$  is the number of time periods; this was improved later in, e.g., see [1, 9, 24]. Polynomial time algorithms have also been developed for variants of ULS, including the constant capacity problem [11, 23], the uncapacitated problem with backlogging [10], and the uncapacitated problem with demand time windows [14].

---

\* Corresponding author.



In many situations the assumption of known, deterministic data (such as demand) is not necessarily realistic. For instance, the demand for each time period is unknown in advance. Significant research has been done on developing optimal inventory policies by assuming that demands are independent and they follow certain distributions (see, for example, [8] for the classical work of  $(s, S)$  policies, and many others). In this paper we assume demands can be dependent between different time periods and study extensions of deterministic lot-sizing problems in which a stochastic programming approach [18] is adopted to address uncertain problem parameters. It is named stochastic uncapacitated lot-sizing problem with backlogging. Examples of applications that have submodels of this form embedded within them include stochastic capacity expansion problems [3], stochastic batch-sizing problems [15], and stochastic production planning problems [6].

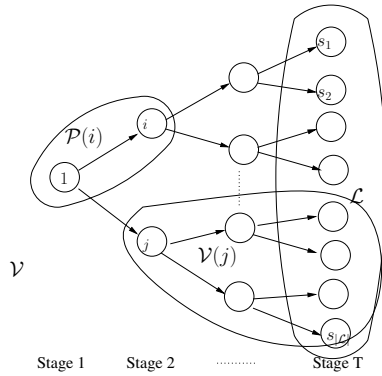
In [2] it is shown that the Wagner-Whitin optimality conditions do not hold even for the version of the problem without backlogging, which makes it non-trivial to derive polynomial time algorithms similar to the deterministic case. Moreover, stochastic ULS with backlogging and related problems are not fixed charge network flow problems (as are the deterministic variants), as will be seen. There is nevertheless clearly significant combinatorial structure present in these models, and our algorithms will take advantage of this. To the best of our knowledge, these algorithms are the first exact polynomial time algorithms for stochastic lot-sizing problems with backlogging.

An outline of the paper follows. We introduce the notation and a mathematical formulation of stochastic ULS with backlogging in Section 2. In Section 3, we first state the production path property for stochastic ULS with backlogging, a fundamental optimality condition analogous to the Wagner-Whitin property. We then use this property to develop a characterization of the dynamic programming (DP) value function in terms of breakpoints, breakpoint evaluations, and slopes, which further allows us to show that the DP approach yields a polynomial time algorithm to fully characterize the value function. In Section 4, we discuss how to extend our algorithm to the general tree structured stochastic ULS with backlogging. Finally, we summarize our results in Section 5.

## 2 Notation and Mathematical Formulation

We assume that the uncertain problem parameters evolve as a discrete time stochastic process with finite probability space. The resulting information structure can be interpreted as a scenario tree with  $T$  levels (stages), where node  $i$  in stage  $t$  of the tree gives the state of the system that can be distinguished by information available up to stage  $t$ . Each node  $i$  of the scenario tree, except the root node (indexed as  $i = 1$ ), has a unique parent  $a(i)$ .

Note that the resulting problem, while having significant combinatorial structure, is not a fixed charge network flow problem, since the same inventory that “flows” out of a given node  $i$  must flow to each of  $i$ ’s children. Let  $\mathcal{V}(i)$  represent the set of all descendants of node  $i$  (including  $i$  itself); let  $\mathcal{V} = \mathcal{V}(1)$  represent



**Fig. 1.** Multi-stage stochastic scenario tree formulation

the set of all nodes in the tree and the total number of nodes in the tree is  $n$  (i.e.,  $n = |\mathcal{V}|$ ). The set of leaf nodes is denoted by  $\mathcal{L}$  and the set of nodes on the path from the root node to node  $i$  is denoted by  $\mathcal{P}(i)$ . If  $i \in \mathcal{L}$ , then  $\mathcal{P}(i)$  corresponds to a *scenario* and represents a joint realization of the problem parameters over all periods  $1, \dots, T$ . Let the probability associated with the state represented by node  $i$  be  $p_i$ , and let  $t(i)$  denote the time stage or level of node  $i$  in the tree; i.e.,  $t(i) = |\mathcal{P}(i)|$  and  $t(i) = T$  for each  $i \in \mathcal{L}$ . Let  $\mathcal{C}(i)$  denote the children of node  $i$ , i.e.,  $\mathcal{C}(i) = \{j \in \mathcal{V} : a(j) = i\}$ . We will initially assume that the scenario tree is balanced, i.e., that there is a fixed number of children for each non-leaf node. We then extend our results to other cases. Throughout we will let  $\mathcal{C} = \max_{i \in \mathcal{V}}\{|\mathcal{C}(i)|\}$  (and therefore  $|\mathcal{C}(i)| = \mathcal{C}$  for all  $i \in \mathcal{V} \setminus \mathcal{L}$  in a balanced tree.) We can now express a multi-stage stochastic MIP formulation of stochastic ULS with backlogging as

$$\begin{aligned}
 \min \quad & \sum_{i \in \mathcal{V}} (\alpha_i x_i + \beta_i y_i + h_i s_i^+ + b_i s_i^-) \\
 \text{s.t.} \quad & s_{a(i)}^+ + s_i^- + x_i = d_i + s_i^+ + s_{a(i)}^- \quad \forall i \in \mathcal{V}, \tag{1} \\
 & x_i \leq M y_i \quad \forall i \in \mathcal{V}, \tag{2} \\
 & x_i, s_i^+, s_i^- \geq 0, y_i \in \{0, 1\} \quad \forall i \in \mathcal{V}. \tag{3}
 \end{aligned}$$

Decision variables  $x_i$ ,  $s_i^+$ , and  $s_i^-$  represent the levels of production, inventory, and backlogging at the end of period  $t(i)$  corresponding to the state defined by node  $i$ , and  $y_i$  is the setup variable for node  $i$ . Parameters  $\alpha_i$ ,  $\beta_i$ ,  $h_i$ ,  $b_i$ , and  $d_i$  represent unit production, setup, holding, backlogging costs, and the demand in node  $i$ . We assume that all of these parameters are nonnegative and may be stochastic in nature. For each  $i \in \mathcal{L}$ , we further assume that  $\alpha_i < b_i$ , as otherwise production will never occur at that node. For notational brevity, probability  $p_i$  is included in problem parameters  $\alpha_i$ ,  $\beta_i$ ,  $h_i$ , and  $b_i$ . Constraint (1) represents the inventory flow balance and constraint (2) indicates the relationship between production and setup for each node  $i$ . For any node  $i \in \mathcal{V}$  and any node  $j \in \mathcal{V}(i)$ ,

define  $d_{ij} = \sum_{n \in \mathcal{P}(j) \setminus \mathcal{P}(a(i))} d_n$ . Note that since demand may be quite high in some scenarios, allowing the possibility of backlogging may be necessary to ensure that the problem has a feasible solution. Since at most one of  $s_i^+$  and  $s_i^-$  will be positive in an optimal solution, for all  $i \in \mathcal{V}$ , it will be convenient throughout to refer to  $s_i = s_i^+ - s_i^-$ . Thus, if  $s_i > 0$ , it represents the level of inventory; and if  $s_i < 0$ , then  $|s_i|$  represents the quantity of backlogging.

### 3 A Dynamic Programming Framework

We first introduce the production path property, which defines optimality conditions for stochastic ULS with backlogging. This will allow us to define a backward recursion for which the value function of each node  $i$  is piecewise linear and continuous; these value functions can therefore be analyzed in terms of breakpoints (i.e., points in the domain at which slope change occurs), functional evaluations of breakpoints, and the slopes of the segments to the right of the breakpoints. We achieve the complexity bound by analyzing the number of breakpoints whose evaluations and right slopes must be stored and computed at each node, and by analyzing how long these computations take. For both of these calculations the production path property will be essential.

Due to space limitations, many of the proofs are abbreviated and/or omitted. Some proofs for analogous results for stochastic ULS *without* backlogging can be found in [13]; however, to the best of our knowledge, this document contains the first algorithmic analysis of stochastic programming models with backlogging (such models are *not* studied in [13]).

**Proposition 1.** (*Production Path Property*) *For any instance of stochastic ULS with backlogging, there exists an optimal solution  $(x^*, y^*, s^*)$  such that for each node  $i \in \mathcal{V}$ ,*

$$\text{if } x_i^* > 0, \text{ then } x_i^* = d_{ik} - s_{a(i)}^* \text{ for some } k \in \mathcal{V}(i). \tag{4}$$

In other words, there always exists an optimal solution such that if we produce at a node  $i$ , then we produce exactly enough to satisfy demand along the path from node  $i$  to some descendant of node  $i$ . This can be proven by showing that any optimal solution that contains a node that violates (4) is a convex combination of optimal solutions, at least one of which contains one less node that violates (4).

The production path property clearly implies there are  $n$  candidates for  $x_1$  (production at the root node) in an optimal solution and therefore we have immediately

**Proposition 2.** *There exists an algorithm that runs in linear time for stochastic ULS with backlogging with two periods.*

For the multi-period stochastic ULS with backlogging, let  $\mathcal{H}(i, s)$  represent the optimal value function for node  $i \in \mathcal{V}$  when the inventory left from previous period is  $s$  (for notational brevity, we will use  $s$  rather than  $s_{a(i)}$  for the second argument of this function).

For each node  $i \in \mathcal{V}$  we have two options: production or non-production. If production occurs, then the value function for this node contains 1) setup, production and inventory costs corresponding to this node and 2) the cost for later periods. (Note that we will produce no less than what is required to satisfy demand in at least the current period, and so we will not incur backlogging costs if we produce.) We will call this function the *production value function*  $\mathcal{H}_P(i, s)$ . From the production path property, the production quantity at node  $i$  is  $x_i = d_{ik} - s$  for some  $k \in \mathcal{V}(i)$  such that  $d_{ik} > s$ . Therefore

$$\mathcal{H}_P(i, s) = \beta_i + \min_{k \in \mathcal{V}(i): d_{ik} > s} \left\{ \alpha_i(d_{ik} - s) + h_i(d_{ik} - d_i) + \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, d_{ik} - d_i) \right\}. \tag{5}$$

Otherwise, if no production occurs, then the value function for this node contains only 1) either inventory holding costs or backlogging costs corresponding to this node, depending on whether  $s - d_i$  is positive or negative, and 2) the cost for later periods. We will call this function the *non-production value function*  $\mathcal{H}_{NP}(i, s)$ . This function can be expressed as

$$\mathcal{H}_{NP}(i, s) = \max \{h_i(s - d_i), -b_i(s - d_i)\} + \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s - d_i). \tag{6}$$

Note that we can only exclude the possibility of production at  $i$  if  $s \geq \max_{j \in \mathcal{V}(i)} d_{ij}$ . The value function under this condition can be defined  $\mathcal{H}(i, s) = \mathcal{H}_{NP}(i, s)$ . In all other cases (including negative values of  $s$ ), we must consider both production and non-production because of the possibility of backlogging. Therefore for any  $s \leq \max_{j \in \mathcal{V}(i)} d_{ij}$ , the backward recursion function can be described as

$$\mathcal{H}(i, s) = \min \{\mathcal{H}_P(i, s), \mathcal{H}_{NP}(i, s)\}. \tag{7}$$

Let  $[1], \dots, [|\mathcal{V}(i)|]$  be an ordering of the descendants of node  $i$  such that  $d_i = d_{i[1]} \leq d_{i[2]} \leq \dots \leq d_{i[|\mathcal{V}(i)|]}$ . In the remaining part of this section, we study a  $T$  period balanced scenario tree case in which each non-leaf node contains a fixed number of children (i.e.,  $\mathcal{C}$  children for each node in the tree).

**Observation 1.** *The summation of piecewise linear and continuous functions is still a piecewise linear and continuous function.*

**Observation 2.** *The minimum of two piecewise linear, continuous functions is still a piecewise linear and continuous function.*

**Proposition 3.** *The value function  $\mathcal{H}(i, s)$  for each node  $i \in \mathcal{V}$  is piecewise linear and continuous.*

*Proof.* By induction.

**Base case:** In period  $T$ , the value function for each node  $i$  is

$$\mathcal{H}(i, s) = \begin{cases} \alpha_i(d_i - s) + \beta_i & \text{if } s < \underline{d}_i, \\ b_i(d_i - s) & \text{if } \underline{d}_i \leq s < d_i, \\ h_i(s - d_i) & \text{if } s \geq d_i, \end{cases}$$

where

$$\underline{d}_i = d_i - \frac{\beta_i}{b_i - \alpha_i}$$

(Note that  $\underline{d}_i$  is not necessarily nonnegative.) This value function is piecewise linear and continuous as shown in Figure 2.

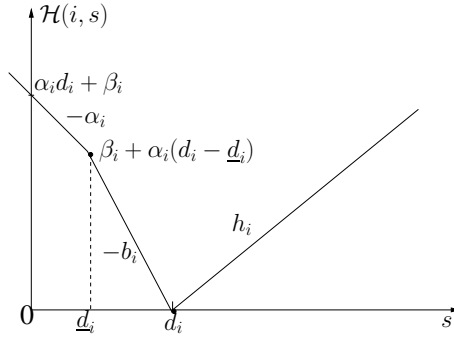


Fig. 2. Value function  $\mathcal{H}(i, s)$  for node  $i$  at time period  $T$

**Inductive step:** It is not difficult to show that  $\mathcal{H}_P(i, s)$  is the minimum of linear functions (see (5)) and is therefore itself piecewise linear and discontinuous with the same slope at each piece. Using the induction hypothesis, it is also not difficult to show that  $\mathcal{H}_{NP}(i, s)$  is the sum of piecewise linear, continuous functions (see (6)) and is therefore itself piecewise linear and continuous.

Therefore  $\mathcal{H}(i, s)$  is the minimum of a piecewise linear and continuous function (see (7)) and a piecewise linear and discontinuous function. We can claim that during each piece of the production value function, there exists a point  $s'$  such that  $\mathcal{H}_{NP}(i, s') = \mathcal{H}_P(i, s')$  and  $\mathcal{H}_{NP}(i, s) < \mathcal{H}_P(i, s)$  for each  $s > s'$ . To prove the claim, let  $s^* = d_{ik} - \epsilon$  where

$$k = \operatorname{argmin} \left\{ \alpha_i d_{ik} + h_i(d_{ik} - d_i) + \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, d_{ik} - d_i) \right\}. \quad (8)$$

Then, we discuss it in two cases:

(1) If  $k = i$ , then

$$\begin{aligned} & \mathcal{H}_P(i, s^*) - \mathcal{H}_{NP}(i, s^*) \\ &= \beta_i + \alpha_i \epsilon + \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, 0) \\ & - (b_i \epsilon + \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, -\epsilon)). \end{aligned}$$

Therefore, there exists an  $\epsilon > 0$  such that  $\mathcal{H}_P(i, s^*) \geq 0$  as long as  $\beta_i > 0$ .

(2) If  $k > i$ , then

$$\begin{aligned} & \mathcal{H}_P(i, s^*) - \mathcal{H}_{NP}(i, s^*) \\ &= \beta_i + \alpha_i \epsilon + h_i(d_{ik} - d_i) + \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, d_{ik} - d_i) \\ & - (h_i(d_{ik} - d_i - \epsilon) + \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, d_{ik} - d_i - \epsilon)). \end{aligned}$$

Conclusion also holds. That is, there exists an  $\epsilon > 0$  such that  $\mathcal{H}_P(i, s^*) - \mathcal{H}_{NP}(i, s^*) > 0$ .

Therefore, we have the following conclusion:

Corresponding to each line piece of production value function, we have

$$\mathcal{H}(i, s) = \begin{cases} \min\{\mathcal{H}_P(i, s), \mathcal{H}_{NP}(i, s)\} & \text{if } s < s^* \\ \mathcal{H}_{NP}(i, s), & \text{if } s > s^*, \end{cases} \tag{9}$$

and  $\mathcal{H}_P(i, s^*) = \mathcal{H}_{NP}(i, s^*)$ . Thus, the claim holds and the inductive step is also right. □

This result is interesting in light of the fact that the value function for the case of stochastic ULS in which backlogging is *not* permitted is *not* continuous, only right continuous [13].

**Corollary 1.** *The non-production value function  $\mathcal{H}_{NP}(i, s)$  is piecewise linear and continuous.*

We can let *breakpoints* represent the  $s$  values at which the slope of the value function changes. The value function  $\mathcal{H}(i, s)$  can then be computed and stored in terms of breakpoints, evaluations of breakpoints, and slopes immediately to the right of the breakpoints. Let  $B(i)$  represent the set of breakpoints for the value function  $\mathcal{H}(i, s)$  at node  $i$ . The complexity of the problem depends on the number of breakpoints.

**Proposition 4.** *The number of breakpoints corresponding to the value function  $\mathcal{H}(i, s)$  for each node  $i \in \mathcal{V}$  is bounded by  $\mathcal{O}(|\mathcal{V}(i)|^{\log_2(3/2)+1})$ .*

*Proof.* We calculate the number of breakpoints  $|B(i)|$  for each node  $i \in \mathcal{V}$  backwards from time period  $T$  to time period 1. First, as shown in Figure 2, there are three breakpoints for the value function corresponding to each leaf node (i.e., including  $s = -\infty$ ). To calculate the number of breakpoints  $|B(i)|$  for each node  $i \in \mathcal{V}$ , we consider the number of breakpoints generated by  $\mathcal{H}_{NP}(i, s)$  and  $\mathcal{H}_P(i, s)$ , respectively. We can show that the number of breakpoints for  $\mathcal{H}_{NP}(i, s)$  is no more than the total number of breakpoints generated by  $\mathcal{H}(\ell, s)$  for each  $\ell \in \mathcal{C}(i)$ . Thus the number of breakpoints for  $\mathcal{H}_{NP}(i, s)$  is

$$|B_{NP}(i)| \leq \sum_{\ell \in \mathcal{C}(i)} |B(\ell)| + 1 - 1 = \sum_{\ell \in \mathcal{C}(i)} |B(\ell)|. \tag{10}$$

Note here that we have one more breakpoint generated by  $s = d_i$  according to the formulation of  $\mathcal{H}_{NP}(i, s)$  and at least one replicated breakpoint  $s = -\infty$ .

For  $\mathcal{H}_P(i, s)$ , we can show from (5) that  $\mathcal{H}_P(i, s)$  is linear in the interval  $d_{i[r-1]} \leq s < d_{i[r]}$ . Therefore, the possible breakpoints for  $\mathcal{H}_P(i, s)$  are  $d_{i[1]}, d_{i[2]}, \dots, d_{i[|\mathcal{V}(i)|]}$ , which are also breakpoints in  $B_{NP}(i)$ . Recall also that  $\mathcal{H}_{NP}(i, s)$  is piecewise linear (Corollary 1) and continuous. We call the breakpoints of  $B_{NP}(i)$  *old* breakpoints for this iteration.

Then we can see from the above observations that the value function  $\mathcal{H}(i, s)$  between two old breakpoints in  $\mathcal{H}_{NP}(i, s)$  can be viewed as the minimum of two linear functions. Since  $\mathcal{H}_{NP}(i, s)$  is piecewise linear and continuous, between any two old breakpoints, there is at most one *new* breakpoint and there are totally no more than  $|B_{NP}(i)|$  new breakpoints.

We also notice that, based on (9), we have  $\mathcal{H}_{NP}(i, d_{i[r]} - \epsilon) < \mathcal{H}_P(i, d_{i[r]} - \epsilon)$ . Therefore, for the non-production value function  $\mathcal{H}_{NP}(i, s)$  corresponding to each linear piece of  $\mathcal{H}_P(i, s)$  and let  $s_1, s_2, \dots, s_{n-1}, s_n$  be the breakpoints, we have  $\mathcal{H}_{NP}(i, s_1) \leq \mathcal{H}_{NP}(i, s_2)$  and  $\mathcal{H}_{NP}(i, s_{n-1}) \geq \mathcal{H}_{NP}(i, s_n)$  except the first piece. Thus, every time when we add two new breakpoints, there will be one old breakpoint deleted. Therefore we have that

$$|B(i)| \leq 3|B_{NP}(i)|/2. \tag{11}$$

Note here for the first two breakpoints for the first piece, we have at most one new breakpoint generated between these two old breakpoints.

Combining (10) and (11), we have

$$|B(i)| \leq 3/2|B_{NP}(i)| \leq 3/2 \sum_{\ell \in \mathcal{C}(i)} |B(\ell)|. \tag{12}$$

Since  $|B(j)| = 3 = 3|\mathcal{V}(j)|$  for each leaf node  $j$  at time period  $T$ , by induction, we have that

$$|B(i)| \leq 3(3/2)^{T-t(i)}|\mathcal{V}(i)|. \tag{13}$$

Since every non-leaf node has at least two children, we have

$$|\mathcal{V}(i)| \geq 1 + 2^1 + \dots + 2^{T-t(i)} = 2^{T-t(i)+1} - 1. \tag{14}$$

Then, combining (13) and (14), the number of breakpoints

$$|B(i)| = 3(2^{T-t(i)})^{\log_2(3/2)}|\mathcal{V}(i)| \leq 3((|\mathcal{V}(i)| + 1)/2)^{\log_2(3/2)}|\mathcal{V}(i)|. \quad \square$$

We can also easily obtain the following corollary

**Corollary 2.** *For a balanced tree case with  $\mathcal{C}(i) = \mathcal{C}$  for each  $i \in \mathcal{V} \setminus \mathcal{L}$ , the number of breakpoints corresponding to the value function  $\mathcal{H}(i, s)$  for each node  $i \in \mathcal{V}$  is bounded by  $\mathcal{O}(|\mathcal{V}(i)|^{\log_2(3/2)+1})$ .*

The DP algorithm is simply a construction of  $\mathcal{H}(1, s)$ , the value function at the root node.

**Theorem 1.** *The value function for stochastic ULS with backlogging can be solved by a dynamic programming algorithm that runs in  $\mathcal{O}(n^{\log_2(3/2)+2} \log \mathcal{C})$  time.*

*Proof.* The value function  $\mathcal{H}(i, s)$  can be stored in terms of breakpoint values, evaluations of breakpoints, and the right slope of each breakpoint. Starting from each leaf node as shown in Figure 2, the value function can be stored in three parts:

- (1) Three breakpoints:  $s = -\infty$ ,  $s = \underline{d}_i = d_i - \frac{\beta_i}{b_i - \alpha_i}$ , and  $s = d_i$ . Note that  $\mathcal{H}_P(i, \underline{d}_i) = \mathcal{H}_{NP}(i, \underline{d}_i)$  and that

$$\frac{\partial \mathcal{H}_P(i, \underline{d}_i)}{\partial s} = -\alpha_i > -b_i = \frac{\partial \mathcal{H}_{NP}(i, \underline{d}_i)}{\partial s}.$$

- (2) Evaluations of the breakpoints  $\mathcal{H}(i, -\infty) = +\infty$ ,  $\mathcal{H}(i, \underline{d}_i) = \alpha_i(d_i - \underline{d}_i) + \beta_i = b_i(d_i - \underline{d}_i)$  and  $\mathcal{H}(i, d_i) = 0$
- (3) The right slope of each breakpoint ( $r(i, \underline{d}_i) = -b_i$ , and  $r(i, d_i) = h_i$ ) and the left slope of the first breakpoint ( $r(i, -\infty) = -\alpha_i$ ).

For the general case to store the value function  $\mathcal{H}(i, s)$  for each node  $i \in \mathcal{V}$ , we need to perform the following calculation steps (we have summarized the bookkeeping of the work associated with each step for space reasons):

- (1) Generate the list of breakpoints of  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$  in non-decreasing order. Since the  $\mathcal{O}(|\mathcal{V}(\ell)|^{\log_2(3/2)+1})$  breakpoints corresponding to each function  $\mathcal{H}(\ell, s)$ ,  $\ell \in \mathcal{C}(i)$ , have already been sorted in non-decreasing order, we obtain the initial non-decreasing sequence  $\kappa$  for the  $\mathcal{C}$  breakpoints corresponding to the first breakpoints in each function  $\mathcal{H}(\ell, s)$ ,  $\ell \in \mathcal{C}(i)$ , which takes  $\mathcal{O}(\mathcal{C} \log \mathcal{C})$  time. Then, to generate each subsequent breakpoint for  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$ , we pick the smallest value breakpoint from  $\kappa$  and add it to the end of the non-decreasing breakpoint list for  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$ . Correspondingly, we add this breakpoint's next one in this breakpoint's original corresponding function  $\mathcal{H}(\ell, s)$  into  $\kappa$ . It takes  $\mathcal{O}(\log \mathcal{C})$  time to find the right position in  $\kappa$  for the newly added breakpoint. Since  $\mathcal{O}(\mathcal{C} \log \mathcal{C}) \leq \mathcal{O}(|\mathcal{V}(i)|^{\log_2(3/2)+1} \log \mathcal{C})$ . Thus, this entire step can be completed in  $\mathcal{O}(|\mathcal{V}(i)|^{\log_2(3/2)+1} \log \mathcal{C})$  time.
- (2) Calculate and store  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$ . We can use Proposition 4 to show that this step can be completed in  $\mathcal{O}(|\mathcal{V}(i)|^{\log_2(3/2)+1})$  time.
- (3) Calculate and store the breakpoints, evaluations of breakpoints, and slopes for the non-production value function  $\mathcal{H}_{NP}(i, s)$ . This step can be completed in  $\mathcal{O}(|\mathcal{V}(i)|^{\log_2(3/2)+1})$  time.
- (4) Calculate  $\rho(k) = \alpha_i d_{ik} + h_i(d_{ik} - d_i) + \sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, d_{ik} - d_i)$  and store it for each  $k \in \mathcal{V}(i)$ .

We can use the production path property to show that this can be completed in  $\mathcal{O}(|\mathcal{V}(i)| \log |\mathcal{V}(i)|)$  time.

- (5) Calculate and store the breakpoints, evaluations of breakpoints, and slopes for the production value function  $\mathcal{H}_P(i, s)$ :

This step can be completed in  $\mathcal{O}(|\mathcal{V}(i)| \log |\mathcal{V}(i)|)$  time.

- (6) Calculate and store the breakpoints, evaluations of breakpoints, and slopes for the value function  $\mathcal{H}(i, s)$ :

This step amounts to defining the *new* breakpoints for  $i$  by comparing  $\mathcal{H}_P(i, s)$  and  $\mathcal{H}_{NP}(i, s)$  within the intervals defined by each consecutive pair



of *old* breakpoints, and it can be completed in  $\mathcal{O}(|\mathcal{V}(i)|^{\log_2(3/2)+1})$  time since the number of breakpoints is bounded by  $\mathcal{O}(|\mathcal{V}(i)|^{\log_2(3/2)+1})$  (from Proposition 4).

The maximum complexity for each step is  $\mathcal{O}(|\mathcal{V}(i)|^{\log_2(3/2)+1} \log \mathcal{C})$ . Since the above operations are required for each node  $i \in \mathcal{V}$ , the optimal value function for the stochastic ULS with backlogging can be obtained and stored in  $\mathcal{O}(n^{\log_2(3/2)+2} \log \mathcal{C})$  time.  $\square$

We can also obtain the following corollary.

**Corollary 3.** *For a balanced tree case with  $\mathcal{C}(i) = \mathcal{C}$  for each  $i \in \mathcal{V} \setminus \mathcal{L}$ , the value function  $\mathcal{H}(i, s)$  for each node  $i \in \mathcal{V}$  can be obtained in  $\mathcal{O}(n^{\log_c(3/2)+2} \log \mathcal{C})$  time.*

### 4 Stochastic Uncapacitated Lot-Sizing Problem with Backlogging

In this section, we consider the general stochastic ULS with backlogging problem (i.e.,  $\mathcal{C}(i) \geq 1$  for each node  $i \in \mathcal{V} \setminus \mathcal{L}$ ). For this case, it is not known if the full characterization of the value function  $\mathcal{H}(i, s)$  for each  $i \in \mathcal{V}$  can be obtained in polynomial time. However, an optimal solution of the general stochastic ULS with backlogging can be obtained in polynomial time. Without loss of generality, we assume zero initial inventory.

With zero initial inventory, according to Proposition 1, we have  $x_1 = d_{1j}$  and  $s_1 = d_{1j} - d_1$  for some  $j \in \mathcal{V}$  in an optimal solution. Then, corresponding to the value function  $\mathcal{H}(\ell, s)$  for each  $\ell \in \mathcal{C}(1)$ , we only need to store the breakpoints  $s = d_{1j} - d_1$  for all  $j \in \mathcal{V}$  and their evaluations. In general, based on (5), (6) and (7), the breakpoints for the value function  $\mathcal{H}(i, s)$  are obtained by moving the breakpoints for the value function  $\mathcal{H}(\ell, s)$  for all  $\ell \in \mathcal{C}(i)$  to the right by  $d_i$  plus new breakpoints generated by minimizing two piecewise linear functions. Thus, in order to store the breakpoints  $s = d_{1j} - d_{1a(i)}$  for all  $j \in \mathcal{V} : d_{1j} - d_{1a(i)} \geq 0$  for node  $i$ , in the calculation of the value functions  $\mathcal{H}(\ell, s)$  for all  $\ell \in \mathcal{C}(i)$ , we only need to store the breakpoints  $s = d_{1j} - d_{1i} = d_{1j} - d_{1a(\ell)}$  for all  $j \in \mathcal{V} : d_{1j} - d_{1a(\ell)} \geq 0$ . Therefore, in the algorithm described in Section 3 to calculate the value function  $\mathcal{H}(i, s)$  for each node  $i \in \mathcal{V}$  backwards starting from leaf nodes, we store the breakpoints  $s = d_{1j} - d_{1a(i)}$  for all  $j \in \mathcal{V} : d_{1j} - d_{1a(i)} \geq 0$  and their evaluations. There are at most  $\mathcal{O}(n)$  breakpoints for each value function  $\mathcal{H}(i, s)$ .

This reduction in the number of breakpoints means that, at each step calculating the value function  $\mathcal{H}(i, s)$  as described in Section 3, we can bound the time the algorithm will take as follows:

- (1) Since we have stored the same set of  $\mathcal{O}(n)$  breakpoints  $s = d_{1j} - d_{1a(\ell)} = d_{1j} - d_{1i}$  for all  $j \in \mathcal{V} : d_{1j} - d_{1a(\ell)} \geq 0$  corresponding to each value function  $\mathcal{H}(\ell, s)$ ,  $\ell \in \mathcal{C}(i)$ , the list of breakpoints of  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$  has the same set of  $\mathcal{O}(n)$  breakpoints. This step takes  $\mathcal{O}(n)$  time.

- (2) Based on Step (1), we have the same set of  $\mathcal{O}(n)$  breakpoints for each  $\mathcal{H}(\ell, s)$ ,  $\ell \in \mathcal{C}(i)$  as well as  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$ . Based on Step (1), we have the same set of  $\mathcal{O}(nT)$  breakpoints for each  $\mathcal{H}(\ell, s)$ ,  $\ell \in \mathcal{C}(i)$  as well as  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$ . This step can be finished by defining a separated list of  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$  for node  $i$  and update the value for each breakpoint of  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$  as a byproduct when obtain the value function  $\mathcal{H}(\ell, s)$  for each  $\ell \in \mathcal{C}(i)$ . It takes  $\mathcal{O}(n)$  time to update the breakpoints for the list  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$  for each  $\ell \in \mathcal{C}(i)$ .
- (3) Based on Step (2), this step can be completed in  $\mathcal{O}(n)$  time since the number of breakpoints is bounded by  $\mathcal{O}(n)$ .
- (4) Based on Step (2), the function values  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, d_{ij} - d_i)$  for each  $j \in \mathcal{V}$  are stored in  $\sum_{\ell \in \mathcal{C}(i)} \mathcal{H}(\ell, s)$  since  $d_{ij} - d_i = d_{1j} - d_{1i} = d_{1j} - d_{1a(\ell)}$  for each  $\ell \in \mathcal{C}(i)$ . There are  $|\mathcal{V}(i)| \leq n$  candidates for  $\rho(k)$  and the value of each  $\rho(k)$  can be obtained by binary search in  $\mathcal{O}(\log n)$  time since the number of breakpoints is bounded by  $\mathcal{O}(n)$ . Therefore, this entire step can be completed in  $\mathcal{O}(n \log n)$  time.
- (5) This step can be completed in  $\mathcal{O}(n \log n)$  time since the number of breakpoints is bounded by  $\mathcal{O}(n)$ .
- (6) To calculate and store the breakpoints of  $\mathcal{H}(i, s)$  and obtain the minimum in this expression, we compare each linear piece of the production value function  $\mathcal{H}_P(i, s)$  with several breakpoint values of the corresponding non-production value function. This step can be completed in  $\mathcal{O}(n)$  time since the number of breakpoints is bounded by  $\mathcal{O}(n)$ .

Therefore, the total amount of work required at each node is bounded by  $\mathcal{O}(n \log n)$ . Since the above operations are required for each node  $i \in \mathcal{V}$ , the following conclusion holds.

**Theorem 2.** *The general stochastic ULS with backlogging can be solved in  $\mathcal{O}(n^2 \log n)$  time.*

The general stochastic ULS with backlogging problem for which the initial inventory level is unknown and is itself a decision variable can be transformed into another general stochastic ULS with backlogging problem with zero initial inventory by adding a dummy root node 0 as the parent node of node 1 with zero production, setup and inventory costs as well as zero demand.

## 5 Conclusions

In this paper, we first studied that the value function  $\mathcal{H}(i, s)$  for each  $i \in \mathcal{V}$  for stochastic ULS with backlogging is piecewise linear and continuous. Correspondingly, for the case  $\mathcal{C}(i) \geq 2$  for each  $i \in \mathcal{V} \setminus \mathcal{L}$ , we developed a polynomial time algorithm that runs in  $\mathcal{O}(n^3 \log C)$  time for obtaining the full characterization of the value function, which was further improved to be  $\mathcal{O}(n^3)$  time for the balanced tree case (i.e.,  $\mathcal{C}(i) = \mathcal{C}, \forall i \in \mathcal{V} \setminus \mathcal{L}$ ). Then, we extended our analysis to a more general tree structure. For instance, there are several nodes  $i \in \mathcal{V} \setminus \mathcal{L}$  such that  $\mathcal{C}(i) = 1$ . In this case, the number of nodes in the tree

can be polynomial to the number of time periods. For this general tree case, we developed a polynomial time algorithm that runs in  $\mathcal{O}(n^2 \log n)$  time. Since stochastic ULS with backlogging is a fundamental stochastic integer programming structure, our studies on stochastic ULS with backlogging will help solve many large complicated stochastic integer programming problems.

Valid inequalities for stochastic ULS and its variant with constant production capacities have been proposed in [12] and [21]. In the future we intend to combine our approaches, polynomial time algorithms and polyhedral studies, into decomposition frameworks for large size stochastic integer programming problems, such as those studied in [7] and [19] (among others).

## Acknowledgments

This research was supported in part by NSF grants CMMI-0700868 and IIP-0725843 for the first author and by Air Force Scientific Research grant FA9550-04-1-0192 and NSF grant DMI-0323299 for the second author. The authors also thank the four referees for their detailed and helpful comments.

## Bibliography

- [1] Aggarwal, A., Park, J.K.: Improved algorithms for economic lot size problems. *Operations Research* 41, 549–571 (1993)
- [2] Ahmed, S., King, A.J., Parija, G.: A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization* 26, 3–24 (2003)
- [3] Ahmed, S., Sahinidis, N.V.: An approximation scheme for stochastic integer programs arising in capacity expansion. *Operations Research* 51, 461–471 (2003)
- [4] Belvaux, G., Wolsey, L.A.: *bc – prod*: a specialized branch-and-cut system for lot-sizing problems. *Management Science* 46, 724–738 (2000)
- [5] Belvaux, G., Wolsey, L.A.: Modelling practical lot-sizing problems as mixed integer programs. *Management Science* 47, 993–1007 (2001)
- [6] Beraldi, P., Ruszczyński, A.: A branch and bound method for stochastic integer problems under probabilistic constraints. *Optimization Methods and Software* 17, 359–382 (2002)
- [7] Carøe, C.C.: Decomposition in stochastic integer programming. PhD thesis, University of Copenhagen (1998)
- [8] Clark, A.J., Scarf, H.: Optimal policies for a multi-echelon inventory problem. *Management Science* 6, 475–490 (1960)
- [9] Federgruen, A., Tzur, M.: A simple forward algorithm to solve general dynamic lot sizing models with  $n$  periods in  $\mathcal{O}(n \log n)$  or  $\mathcal{O}(n)$  time. *Management Science* 37, 909–925 (1991)
- [10] Federgruen, A., Tzur, M.: The dynamic lot-sizing model with backlogging—a simple  $\mathcal{O}(n \log n)$  algorithm and minimal forecast horizon procedure. *Naval Research Logistics* 40, 459–478 (1993)
- [11] Florian, M., Klein, M.: Deterministic production planning with concave costs and capacity constraints. *Management Science* 18, 12–20 (1971)

- [12] Guan, Y., Ahmed, S., Nemhauser, G.L., Miller, A.J.: A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. *Mathematical Programming* 105, 55–84 (2006)
- [13] Guan, Y., Miller, A.J.: Polynomial time algorithms for stochastic uncapacitated lot-sizing problems. *Operations Research* (to appear, 2007)
- [14] Lee, C.Y., Cetinkaya, S., Wagelmans, A.: A dynamic lot-sizing model with demand time windows. *Management Science* 47, 1384–1395 (2001)
- [15] Lulli, G., Sen, S.: A branch-and-price algorithm for multi-stage stochastic integer programming with application to stochastic batch-sizing problems. *Management Science* 50, 786–796 (2004)
- [16] Nemhauser, G.L., Wolsey, L.A.: *Integer and Combinatorial Optimization*. Wiley, New York (1988)
- [17] Pochet, Y., Wolsey, L.A.: *Production Planning by Mixed Integer Programming*. Springer, New York (2006)
- [18] Ruszczyński, A., Shapiro, A. (eds.): *Stochastic Programming*. *Handbooks in Operations Research and Management Science*, vol. 10. Elsevier Science B.V., Amsterdam (2003)
- [19] Sen, S., Sherali, H.D.: Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming* 106, 203–223 (2006)
- [20] Stadtler, H.: Multi-level lot-sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research* 51, 487–502 (2003)
- [21] Di Summa, M., Wolsey, L.A.: Lot-sizing on a tree. Technical report, CORE, UCL, Louvain-la-Neuve, Belgium (2006)
- [22] Tempelmeier, H., Derstroff, M.: A Lagrangean-based heuristic for dynamic multi-level multiitem constrained lot-sizing with setup times. *Management Science* 42, 738–757 (1996)
- [23] van Hoesel, C.P.M., Wagelmans, A.: An  $\mathcal{O}(T^3)$  algorithm for the economic lot-sizing problem with constant capacities. *Management Science* 42, 142–150 (1996)
- [24] Wagelmans, A., van Hoesel, A., Kolen, A.: Economic lot sizing: An  $\mathcal{O}(n \log n)$  algorithm that runs in linear time in the Wagner–Whitin case. *Operations Research* 40, 145–156 (1992)
- [25] Wolsey, L.A.: *Integer Programming*. Wiley, New York (1998)

# Lifting Integer Variables in Minimal Inequalities Corresponding to Lattice-Free Triangles\*

Santanu S. Dey<sup>1</sup> and Laurence A. Wolsey<sup>2</sup>

<sup>1</sup> CORE

Santanu.Dey@uclouvain.be

<sup>2</sup> CORE and INMA,

University Catholique de Louvain,

34, Voie du Roman Pays, 1348 Louvain-la-Neuve, Belgium

laurence.wolsey@uclouvain.be

**Abstract.** Recently, Andersen et al. [1] and Borozan and Cornuéjols [3] characterized the minimal inequalities of a system of two rows with two free integer variables and nonnegative continuous variables. These inequalities are either split cuts or intersection cuts derived using maximal lattice-free convex sets. In order to use these minimal inequalities to obtain cuts from two rows of a general simplex tableau, it is necessary to extend the system to include integer variables (giving the two-dimensional mixed integer infinite group problem), and to develop lifting functions giving the coefficients of the integer variables in the corresponding inequalities. In this paper, we analyze the lifting of minimal inequalities derived from lattice-free triangles.

Maximal lattice-free triangles in  $\mathbb{R}^2$  can be classified into three categories: those with multiple integral points in the relative interior of one of its sides, those with integral vertices and one integral point in the relative interior of each side, and those with non integral vertices and one integral point in the relative interior of each side. We prove that the lifting functions are unique for each of the first two categories such that the resultant inequality is minimal for the mixed integer infinite group problem, and characterize them. We show that the lifting function is not necessarily unique in the third category. For this category we show that a fill-in inequality (Johnson [4]) yields minimal inequalities for mixed integer infinite group problem under certain sufficiency conditions. Finally, we present conditions for the fill-in inequality to be extreme.

## 1 Introduction

Recently, Andersen et al. [1], Borozan and Cornuéjols [3], and Cornuéjols and Margot [5] have developed methods to analyze a system of two rows with two free integer variables and nonnegative continuous variables. They show that facets of system

$$f + \sum_{i=1}^n w^i y_i \in \mathbb{Z}^2 \quad y_i \in \mathbb{R}_+, \quad i \in \{1, \dots, n\} \quad (1)$$

---

\* This text presents research results of the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian State, Prime Minister's Office, Science Policy Programming. The scientific responsibility is assumed by the authors.

are either split cuts or intersection cuts (Balas [2]). These new families of cutting planes are important since some of them cannot be obtained using a finite number of Gomory mixed integer cuts (GMIC); see Cook et al. [4]. We state the following theorem, modified from Borozan and Cornuéjols [3] (See also Theorem 1 in Andersen et al. [1]).

**Theorem 1.** *For the system*

$$f + \sum_{w \in \mathbb{Q}^2} wy(w) \in \mathbb{Z}^2, \quad y(w) \geq 0 \tag{2}$$

where  $y$  has a finite support, an inequality of the form  $\sum_{w \in \mathbb{Q}^2} \pi(w)y(w) \geq 1$  is minimal if the closure of the set

$$P(\pi) = \{w \in \mathbb{Q}^2 | \pi(w - f) \leq 1\} \tag{3}$$

is a maximal lattice-free convex set in  $\mathbb{R}^2$ . Moreover, given a maximal lattice-free convex set  $P(\pi)$  such that  $f \in \text{interior}(P(\pi))$ , the function

$$\pi(w) = \begin{cases} 0 & \text{if } w \in \text{recession cone of } P(\pi) \\ \lambda & \text{if } f + \frac{w}{\lambda} \in \text{Boundary}(P(\pi)) \end{cases} \tag{4}$$

is a minimal valid inequality for [2]. □

One way to obtain valid cutting planes for two rows of a simplex tableau using the minimal inequalities for [2] is to relax the non-basic integer variables to be continuous variables. In order to strengthen such a cutting plane we need to derive minimal inequalities for a system like [2] which also has integer variables. Thus, the goal of this study is to lift integer variables into the minimal inequalities for [2]. This requires characterizing and analyzing valid lifting functions and obtaining the strongest possible coefficients for the integer variables. (See Nemhauser and Wolsey [12] for an overview on lifting).

In Sect. 2 we present the relationship between this lifting problem and minimal inequalities for the mixed integer infinite-group problem. In Sect. 3 we study the fill-in procedure of Johnson [11] and create a framework for analyzing the strength of these inequalities. In Sect(s). 4 and 5 we prove that if  $P(\pi)$  is a triangle with multiple integral points in the relative interior of one side or if  $P(\pi)$  is a triangle with integral vertices and one integral point in the relative interior of each side, then there exists a unique lifting function  $\phi$  such that  $(\phi, \pi)$  is minimal for the mixed integer group problem. In Sect. 6 we prove that if  $P(\pi)$  is a triangle with non-integral vertices with exactly one integral point in the relative interior of each side then there may not exist a unique function  $\phi$  such that  $(\phi, \pi)$  is minimal. We then present sufficient conditions for the fill-in procedure to generate minimal functions in this case. We conclude in Sect. 7.

## 2 Preliminaries

Observe that the integer variables in [2] have no sign restrictions. This corresponds to the so-called group relaxation that was discovered and studied by

Gomory [6], Gomory and Johnson [7, 8, 10, 9] and Johnson [11]. We present notation and a brief overview of mixed integer infinite group problem and establish its relationship to [2].

Let  $I^2$  denote the infinite group of real two-dimensional vectors where addition is taken modulo 1 componentwise, i.e.,  $I^2 = \{(u_1, u_2) \mid 0 \leq u_i < 1 \ \forall 1 \leq i \leq m\}$ . Let  $S^2$  represent the set of real two-dimensional vectors  $w = (w_1, w_2)$  that satisfy  $\max_{1 \leq i \leq 2} |w_i| = 1$ . For an element  $u \in \mathbb{R}^2$ , we use the symbol  $\mathbb{P}(u)$  to denote the element in  $I^2$  whose  $i^{\text{th}}$  entry is  $u_i \pmod{1}$ . We use the symbol  $\bar{0}$  to represent the zero vector in  $\mathbb{R}^2$  and  $I^2$ .

The mixed integer infinite group problem is defined next.

**Definition 1 (Johnson [11]).** *Let  $U$  be a subgroup of  $I^2$  and  $W$  be any subset of  $S^2$ . Then the mixed integer infinite group problem, denoted here as  $MI(U, W, r)$ , is defined as the set of pairs of functions  $x : U \rightarrow \mathbb{Z}_+$  and  $y : W \rightarrow \mathbb{R}_+$  that satisfy*

1.  $\sum_{u \in U} ux(u) + \mathbb{P}(\sum_{w \in W} wy(w)) = r, r \in I^2,$
2.  $x$  and  $y$  have finite supports. □

If all the  $x(u)$ 's are fixed to zero in  $MI(I^2, S^2, r)$ , the problem would reduce to that presented in [2] where  $r \equiv \mathbb{P}(-f)$  [1]. Thus, we need to lift the integer variables into the inequality  $\pi$  to obtain a pair of functions  $(\phi, \pi)$  corresponding to valid inequalities for  $MI(I^2, S^2, r)$ . We next define these valid inequalities for  $MI(I^2, S^2, r)$  more precisely.

**Definition 2 (Johnson [11]).** *A valid function for  $MI(I^2, S^2, r)$  is defined as a pair of functions,  $\phi : I^2 \rightarrow \mathbb{R}_+$  and  $\mu_\phi : S^2 \rightarrow \mathbb{R}_+$ , such that  $\sum_{u \in I^2} \phi(u)x(u) + \sum_{w \in S^2} \mu_\phi(w)y(w) \geq 1, \forall (x, y) \in MI(I^2, S^2, r)$ , where  $\phi(\bar{0}) = 0$  and  $\phi(r) = 1$ . □*

[We will use the terms valid inequality and valid function interchangeably]. Note here that the relationship between the functions  $\pi$  of Theorem [1] and  $\mu_\phi : S^2 \rightarrow \mathbb{R}_+$  in Definition [2] is straight forward. Since  $\pi$  is positively homogenous, we can construct  $\mu_\phi$  in a well-defined fashion by restricting the domain of  $\pi$  to  $S^2$ . Conversely, given  $\mu_\phi, \pi$  is the gauge function which is the homogenous extension of  $\mu_\phi$ . Because of this close relationship, we will use the same symbol for both the functions. See Gomory and Johnson [9] for a presentation of how these inequalities can be used to generate valid cutting planes for two rows of a simplex tableau. Next we define the notion of minimal inequalities.

**Definition 3 (Johnson [11]).** *A valid function  $(\phi, \pi)$  is minimal for  $MI(U, W, r)$  if there does not exist a valid function  $(\phi^*, \pi^*)$  for  $MI(U, W, r)$  different from  $(\phi, \pi)$  such that  $\phi^*(u) \leq \phi(u) \ \forall u \in U$  and  $\pi^*(w) \leq \pi(w) \ \forall w \in W$ . □*

---

<sup>1</sup> Note here that columns corresponding to the continuous variables are assumed to be rational in [2]. However, we will assume that  $W = S^2$ . The function  $\pi$  can be computed for irrational values using Theorem [1]. Therefore, this assumption does not pose any difficulties.

Therefore, given the function  $\pi : \mathbb{R}^2 \rightarrow \mathbb{R}_+$ , the goal of this study is to derive a function  $\phi : I^2 \rightarrow \mathbb{R}_+$  so that the pair  $(\phi, \pi)$  forms minimal inequality for  $MI(I^2, W, r)$ . We next present a proposition (that uses Lemma 5 of Andersen et al. [1] for its proof) to classify maximal lattice-free triangles in  $\mathbb{R}^2$ . Each category is separately analyzed in Sect. 4-6.

**Proposition 1.** *If  $P$  is a maximal lattice-free triangle in  $\mathbb{R}^2$ , then exactly one of the following is true:*

1. *One side of  $P$  contains more than one integral point in its relative interior.*
2. *All the vertices are integral and each side contains one integral point in its relative interior.*
3. *The vertices are non-integral and each side contains one integral point in its relative interior.* □

### 3 Coefficient for Integer Variables: Fill-In Procedure

We begin this Sect. with a presentation of the fill-in procedure developed by Gomory and Johnson [8] and Johnson [11] that is used to generate valid inequalities for the infinite group problem. We then present some techniques to analyze the minimality of the inequalities that are constructed using the fill-in procedure. Finally, we present conditions under which the fill-in inequalities are extreme.

Definition 4 is adapted from the original definition of fill-in procedure.

**Definition 4 (Fill-in procedure).** *Let  $P(\pi)$  be a bounded lattice-free convex set. Let  $G$  be a subset of  $I^2$  such that the subgroup of  $I^2$  generated by  $G$  is finite. Let  $V : G \rightarrow \mathbb{R}_+$  be a function such that*

$$\sum_{u \in G} x(u)V(u) + \sum_{w \in S^2} y(w)\pi(w) \geq 1 \tag{5}$$

*is satisfied for every  $(x, y) \in MI(I^2, S^2, r)$  such that  $x(u) = 0 \ \forall u \notin G$ . Define the fill-in function  $\phi^{G,V} : I^2 \rightarrow \mathbb{R}_+$  as follows:*

$$\phi^{G,V}(u) = \min_{n(v) \in \mathbb{Z}_+} \left\{ \sum_{v \in G} n(v)V(v) + \pi(w) \mid \sum_{v \in G} n(v)v + w \equiv u \right\}. \tag{6}$$

□

It can be verified that the construction of  $\phi^{G,V}$  is equivalent to the original fill-in procedure of Johnson [11] in which we start with the subgroup  $S^G$  of  $I^2$  generated by  $G$  and the valid function  $\check{\phi} : S^G \rightarrow \mathbb{R}_+$  defined as  $\check{\phi}(v) = \min_{n(u) \in \mathbb{Z}_+} \{ \sum_{u \in G} n(u)V(u) \mid \sum_{u \in G} n(u)u = v \}$ . It is easily verified that the pair  $(\phi^{G,V}, \pi)$  forms a subadditive valid inequality for  $MI(I^2, S^2, r)$ .

The fill-in procedure may be interpreted as a two-step lifting scheme. In the first step we obtain the inequality (5) by lifting integer variables corresponding to columns in the set  $G$ . The lifting coefficients, (i.e.,  $V$ ) may depend on the



order of lifting of these variables, i.e., for a given set  $G$  there may exist two different functions  $V_1$  and  $V_2$  such that both functions eventually yield strong cutting planes for  $MI(I^2, S^2, r)$ . Once the integer variables corresponding to columns in the set  $G$  are lifted, the lifting in the second step (of the rest of the integer variables) is completely defined by the choice of  $G$  and  $V$ .

It can be verified that the function  $\phi^{G,V}$  can be evaluated in finite time for each  $u \in I^2$ . We next study conditions under which  $\phi^{G,V}$  is a minimal function. We begin with some results regarding  $\pi$  and  $\phi^{\{\bar{0}\},\{0\}}$  (We denote  $\phi^{\{\bar{0}\},\{0\}}$  by  $\phi^{\bar{0}}$ ).

**Proposition 2.** *If  $\pi$  satisfies at least one point at equality for (2), then  $\phi^{\bar{0}}(r) = 1$ . □*

We next define a subset of  $I^2$  for which we are guaranteed ‘good’ coefficients even if the set  $G$  only contains the element  $\bar{0}$ . This result helps in proving that under certain conditions the lifting function is unique.

**Definition 5.** *Let  $d^1, d^2, \dots$  be the extreme rays of  $P(\pi)$ , i.e.,  $d^i + f$  are the extreme points of  $P(\pi)$ . Let  $s_i$  be the line segment between vertices  $d^i + f$  and  $d^{i+1} + f$  (where  $d^4 := d^1$  when  $P(\pi)$  is triangle). Let  $p^i$  be the set of integer points in the relative interior of  $s_i$ . For an integral point  $X^j \in p^i$ , let  $\delta^{ij}d^i + (1 - \delta^{ij})d^{i+1} + f = X^j$  where  $0 < \delta^{ij} < 1$ . Define the  $D_{ij}(\pi) = \{\rho d^i + \gamma d^{i+1} \mid 0 \leq \rho \leq \delta^{ij}, 0 \leq \gamma \leq 1 - \delta^{ij}\}$ . Let  $D(\pi) = \cup_{i,j} D_{ij}(\pi)$ . □*

See Fig(s). (2) and (3) for illustration of  $D(\pi)$  (represented as the shaded region within the triangle).

**Proposition 3.** *Let  $P(\pi)$  be a bounded maximal lattice free convex set. For any  $v \in D(\pi)$  the following are true:*

1.  $\sum_{u \in I^2} ux(u) + \sum_{w \in \mathbb{R}^2} wy(w) + f \in Z^2$  has a solution  $(\bar{x}, \bar{y})$  with  $\bar{x}(\mathbb{P}(v)) > 0$  which satisfies the cutting plane  $(\phi^{\bar{0}}, \pi)$  at equality.
2.  $\phi^{\bar{0}}(\mathbb{P}(v)) = \pi(v)$ .
3.  $\phi^{\bar{0}}(\mathbb{P}(v)) + \phi^{\bar{0}}(\mathbb{P}(r - v)) = 1$ .
4. If  $(\phi, \pi)$  is any valid inequality for  $MI(I^2, S^2, r)$ , then  $\bar{\phi}(\mathbb{P}(v)) \geq \phi^{\bar{0}}(\mathbb{P}(v))$ . □

**Corollary 1.** *Let  $P(\pi)$  be a bounded maximal lattice free convex set. Then  $\lim_{h \rightarrow 0^+} \frac{\phi^{\bar{0}}(\mathbb{P}(wh))}{h} = \pi(w) \forall w \in \mathbb{R}^2$ . □*

Similar to the proof of Proposition (3) the following result can be proven.

**Proposition 4.** *Let  $P(\pi)$  be a maximal lattice-free triangle. If  $u^* \notin D(\pi)$  and  $\phi^{\bar{0}}(\mathbb{P}(u^*)) = \pi(u^*)$ , then  $\phi^{\bar{0}}(\mathbb{P}(u^*)) + \phi^{\bar{0}}(\mathbb{P}(r - u^*)) > 1$ . □*

Using a characterization for minimal inequalities for  $MI(I^2, S^2, r)$  from Johnson (11) (Theorem 6.1), Proposition (2), Corollary (1), Proposition (4), and the fact that  $\phi^{G,V} \leq \phi^{\bar{0}}$  we can now verify the following result.

**Corollary 2.** *If  $\pi$  is a valid and minimal function for (2) and  $\phi^{G,V}(u) + \phi^{G,V}(r - u) = 1 \forall u \in I^2$ , then  $(\phi^{G,V}, \pi)$  is minimal for  $MI(I^2, S^2, r)$ . Moreover,  $(\phi^{\bar{0}}, \pi)$  is minimal for  $MI(I^2, S^2, r)$  iff  $\mathbb{P}(D(\pi)) = I^2$ . □*

If the function  $\phi^{G,V}$  is minimal, then we show that it must be the unique minimal function. The proof uses the fact that minimal functions must be subadditive. This result allows us to construct extreme inequalities for the infinite group problem using the modified fill-in procedure.

**Theorem 2.** *Let  $(\phi^{G,V}, \pi)$  be minimal for  $MI(I^2, S^2, r)$ . If  $(\phi', \pi)$  is any valid minimal function for  $MI(I^2, S^2, r)$  such that  $\phi'(u) = V(u) \forall u \in G$ , then  $\phi'(v) = \phi^{G,V}(v) \forall v \in I^2$ .  $\square$*

We say that  $(V, \pi)$  is an extreme inequality if  $(V_1, \pi_1)$  and  $(V_2, \pi_2)$  are valid inequalities like (5) and  $V = \frac{1}{2}V_1 + \frac{1}{2}V_2, \pi = \frac{1}{2}\pi_1 + \frac{1}{2}\pi_2$  imply that  $V_1 = V_2 = V, \pi_1 = \pi_2 = \pi$ .

**Corollary 3.** *If  $(V, \pi)$  is extreme and  $(\phi^{G,V}, \pi)$  is minimal for  $MI(I^2, S^2, r)$ , then  $(\phi^{G,V}, \pi)$  is an extreme valid inequality for  $MI(I^2, S^2, r)$ .  $\square$*

### 4 Multiple Integral Points in the Relative Interior of One Side

We begin with a construction that has properties similar to that of a maximal lattice-free triangle with multiple integral points in the relative interior of one side. We denote the length of a line segment  $uv$  as  $|uv|$ . An illustration of Construction 1 is given in Fig. 1.

**Construction 1.** Let  $abc$  be any nontrivial triangle, i.e.,  $a, b$ , and  $c$  are distinct points. Let  $d$  be any point in the interior of the triangle. Let  $ew$  be a line segment

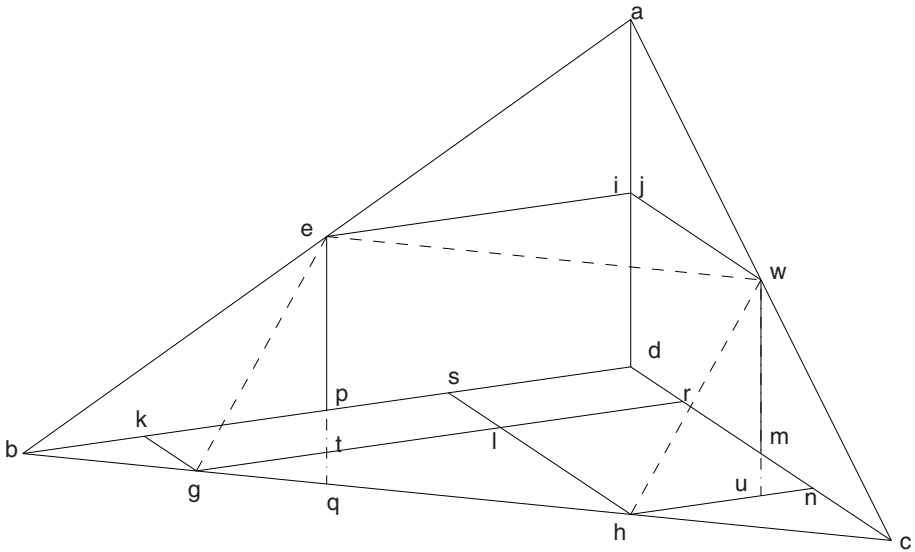


Fig. 1. An example of Construction 1

parallel to the side  $bc$  where  $e$  belongs to the relative interior of line segment  $ab$  and  $w$  belongs to the relative interior of the line segment  $ac$ . Let  $ep$  and  $wm$  be line segments parallel to  $ad$ , meeting  $bd$  and  $cd$  at  $p$  and  $m$  respectively. This is a well-defined step, since  $bda$  and  $cda$  form non trivial triangles.

Let  $q$  be the point at which the line passing through  $ep$  meets  $bc$ . Let  $gh$  be a line segment on  $bc$  such that it has the same length as that of  $ew$ . (This is well-defined since  $e$  and  $w$  belong to the relative interior of lines  $ab$  and  $ac$ . Therefore  $|gh| = |ew| < |bc|$ ). WLOG of generality, we assume that  $g$  lies to the left of  $q$ . (The proof will be similar for the case when  $g$  is to the right of  $q$ ).

Let  $ei$  be a line segment parallel to  $bd$  meeting  $ad$  at  $i$ . Let  $wj$  be a line segment parallel to  $cd$  meeting  $ad$  at  $j$ . Let  $kg$  be a line parallel to  $dc$  meeting  $bd$  at  $k$ . Let  $gr$  be a line segment parallel to  $bd$  meeting  $dc$  at  $r$ . Let  $t$  be the point at which the line passing through  $ep$  meets  $gr$ . Let  $hn$  be a line segment parallel to  $bd$  meeting  $dc$  at  $n$ . Let  $hs$  be a line segment parallel to  $dc$  meeting  $bd$  at  $s$ . (It can be verified that these construction steps are well-defined). Let  $l$  be the point of intersection of  $gr$  and  $hs$ . [This is well-defined: Since  $d$  is in the strict interior of the triangle  $abc$ ,  $bd$  and  $cd$  are not parallel. Therefore the lines passing through the segments  $gr$  and  $hs$  are not parallel and must intersect.]  $\square$

**Proposition 5.** For Construction [1](#),

1.  $i$  and  $j$  are the same point. There exists a point  $u$  such that  $wm$  extended to  $wu$  intersects  $hn$ .
2. Triangle  $glh$  is symmetric to triangle  $eiw$ . [Two triangles are symmetric if the length of their three sides are equal]
3. Triangle  $wuh$  is symmetric to  $etg$ .  $\square$

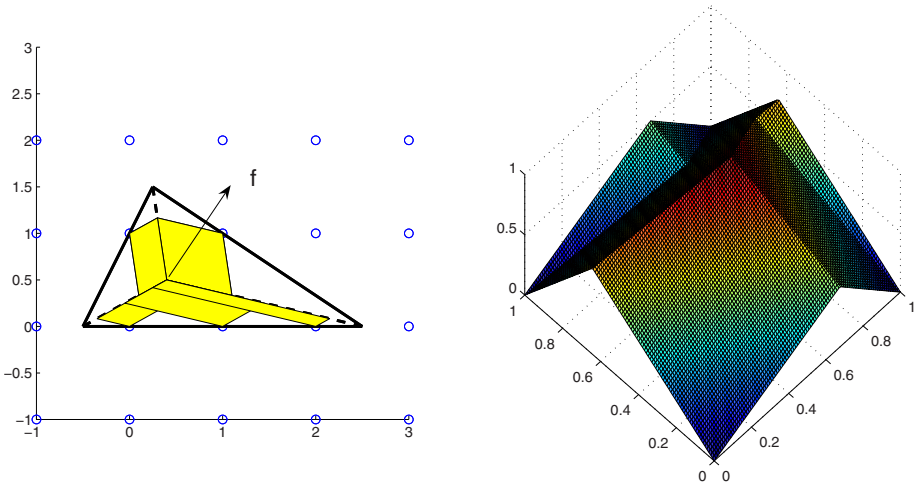
Using Proposition [5](#), we can prove the main result of this Sect., that is presented next.

**Theorem 3.** If  $P(\pi)$  is a maximal lattice-free triangle with multiple integral points in the relative interior of one side, then  $(\phi, \pi)$  is minimal for  $MI(I^2, S^2, r)$  iff  $\phi = \phi^{\bar{0}}$ .  $\square$

This result is similar to the result in one dimension where the unique lifting function yields GMIC. We sketch the main steps in the proof of Theorem [3](#):

1. Any maximal lattice-free triangle with multiple integral points in the relative interior of one side can be represented by the triangle  $abc$  in Construction [1](#), where the point  $d$  represents  $f$  (this is the fractional vector in [\(2\)](#)) and the points  $e, w, g, h$  represent the integral points in the relative interior of each side.
2. Using (2.) and (3.) of Proposition [5](#), it can be verified that  $\mathbb{P}(D(\pi)) = \mathbb{P}(ewgh)$ . Finally, it can be shown that  $\mathbb{P}(ewgh) = I^2$  since  $ew$  and  $gh$  must be parallel and  $e, w, g, h$  represent integral points. The result now follows from Corollary [2](#).

Figure [2](#) shows a maximal lattice-free triangle with 3 integral points in the relative interior of one of its sides ( $D(\pi)$  is represented as the shaded region) and the function  $\phi^{\bar{0}}$ .



**Fig. 2.** Lattice-free Triangle with more than one integral point in the relative interior of one side

### 5 Single Integral Point in the Relative Interior of Each Side and Integral Vertices

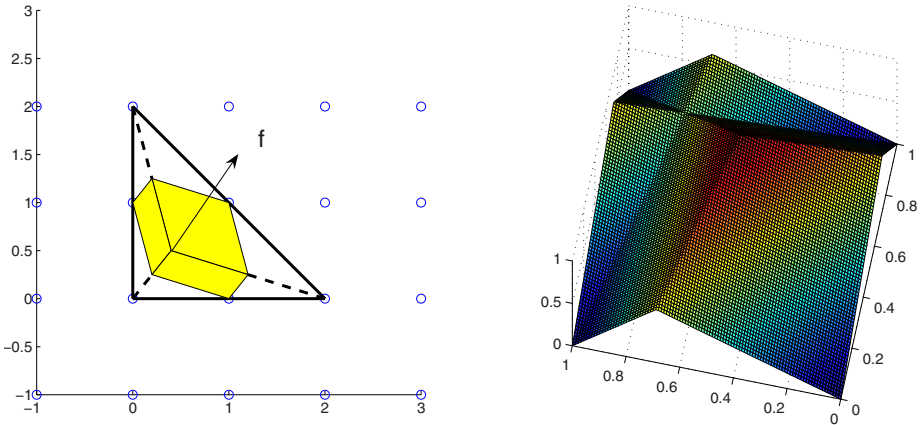
We begin this Sect. with the observation that since (2) is a modular equation, without loss of generality, we may replace  $f$  with  $f + (n_1, n_2)$ , where  $n_1, n_2 \in \mathbb{Z}$  are such that one of the integral points in the relative interior of the side of the triangle is in  $(0, 0)$ . The next proposition shows that a study of a specific subclass of triangles allows us to generalize results to any triangle in this category.

**Proposition 6.** *Let  $P(\pi)$  be a maximal lattice-free triangle with  $f \in \text{interior } P(\pi)$ . Let  $M$  be a two-by-two unimodular matrix. Let  $MP(\pi) = \{(x_1, x_2) | M^{-1}(x_1, x_2) \in P(\pi)\}$ . Define the functions  $M\phi : I^2 \rightarrow \mathbb{R}_+$  and  $M\pi : \mathbb{R}^2 \rightarrow \mathbb{R}_+$  as  $M\phi(u) = \phi(\mathbb{P}(M^{-1}u))$  and  $M\pi(w) = \pi(M^{-1}w)$ . (Note that  $M\pi$  is the function corresponding to the maximal lattice free triangle  $MP(\pi)$  and  $Mf \in MP(\pi)$ .) Then  $(\phi, \pi)$  is a minimal inequality for  $MI(I^2, S^2, r)$  iff  $(M\phi, M\pi)$  is a minimal inequality for  $MI(I^2, S^2, \mathbb{P}(Mr))$ .  $\square$*

We have assumed that one of the integral points in the relative interior of the side of the triangle is in  $(0, 0)$ . Now by applying a suitable unimodular matrix transformation, we can assume without loss of generality that the other two integral point in the relative interior of the other two sides of the triangle are  $(1, 0)$ , and  $(0, 1)$ . Now using a proof similar to that of Theorem 3 and using Proposition 6 we can prove the following result.

**Theorem 4.** *If  $P(\pi)$  is a maximal lattice-free triangle with integral vertices and one integral point in the relative interior of each side, then  $(\phi, \pi)$  is minimal for  $MI(I^2, S^2, r)$  iff  $\phi = \phi^{\bar{0}}$ .  $\square$*

Figure 3 show an example of a lattice-free triangle and  $\phi^{\bar{0}}$  is this case.



**Fig. 3.** Lattice-free triangle with integral vertices and one integral point in the relative interior of each side

### 6 Single Integral Point in the Relative Interior of Each Side and Non-integral Vertices

In this Sect., we first show that in contrast to the previous cases, if  $P(\pi)$  is a triangle with single integral point in the relative interior of each side and non-integral vertices, then  $(\phi^0, \pi)$  is not minimal. We then present some sufficient conditions for generating a minimal inequality using the fill-in procedure.

By Proposition 6, we need to only analyze triangles whose integral points in the relative interior of its sides are:  $(0, 0)$ ,  $(1, 0)$ , and  $(0, 1)$ . Let  $s_1$ ,  $s_2$  and  $s_3$  be the sides of  $P(\pi)$  passing through  $(1, 0)$ ,  $(0, 1)$ , and  $(0, 0)$  respectively. It can be verified that either the slope of  $s_1$  is positive (and  $s_1$  is not vertical) and the slope of  $s_2$  is negative or vice-versa. Henceforth we assume WLOG that slope of  $s_1$  is negative and the slope of  $s_2$  is positive (and  $s_1$  is not vertical).

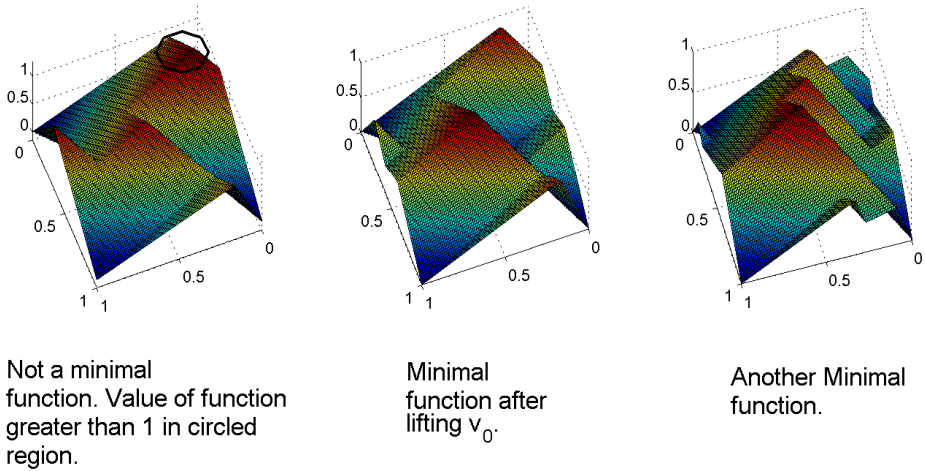
To prove that  $(\phi^0, \pi)$  is not minimal, we show that  $\mathbb{P}(D(\pi))$  is a proper subset of  $I^2$  and then use Corollary 2. This is achieved by verifying that the area of  $D(\pi)$  is less than 1 in this case.

**Proposition 7.** *The area of  $D(\pi)$  is maximized if  $f$  is one of the vertices of  $P(\pi)$ . □*

Now using Propositions 7 and 6 and checking for the maximum possible area of  $D(\pi)$  we obtain the following result.

**Theorem 5.** *If  $P(\pi)$  is a lattice-free triangle with single integral point in the relative interior of each side and non-integral vertices, then  $(\phi^0, \pi)$  is not minimal for  $MI(I^2, S^2, r)$ . □*

*Example 1.* Let  $P(\pi)$  be the triangle with vertices  $(0.25, 1.25)$ ,  $(-0.75, 0.25)$ , and  $(1.25, -5/12)$  and let  $f = (0.5, 0.5)$ . Then it can be verified that  $P(\pi)$  is a



**Fig. 4.** Example where the function  $\phi^{\bar{0}}$  is not minimal.  $\phi^{v_0, V_{v_0}}$  is minimal. There exist distinct functions  $\phi^{v_0}$  and  $\phi_2$  such that  $(\phi^{v_0}, \pi)$  and  $(\phi_2, \pi)$  are minimal.

lattice-free triangle with only one integer point in the relative interior of each of its sides and non-integral vertices.  $\phi^{\bar{0}}(0.1, 0.2) = 1.1$  and  $\phi^{\bar{0}}$  is not minimal. Also there are two distinct functions  $\phi^{v_0}$  (see notation/result of Theorem 6) and  $\phi_2$  such that both  $(\phi^{v_0}, \pi)$  and  $(\phi_2, \pi)$  are minimal. See Fig. 4. □

Therefore, starting from different sets  $G$  and corresponding functions  $V$ , it may be possible to construct different functions  $\phi^{G, V}$  that are all minimal. We first concentrate on the case when  $G$  is a single non-zero element set. In this specific case, we use the notation  $\phi^u$  to denote  $\phi^{\{u\}, V(u)}$  where  $V(u) = \max_{n \in \mathbb{Z}, n \geq 1} \left\{ \frac{1 - \pi(w)}{n} \mid w \equiv r - nu \right\}$ .

The main result on this Sect. is presented next. (This result can be used for any lattice-free maximal triangle with non-integral vertices and only one integral point in the relative interior of each side by using Proposition 6).

**Theorem 6.** *Let  $P(\pi)$  be a triangle whose integral points in the relative interior of its sides are:  $(0, 0)$ ,  $(1, 0)$ , and  $(0, 1)$ . Let the slope of  $s_1$  be negative (and not vertical) and the slope of  $s_2$  be positive. Define  $0 < \delta^{12}, \delta^{23}, \delta^{31} < 1$  such that  $\delta^{12}d^1 + (1 - \delta^{12})d^2 + f = (1, 0)$ ,  $\delta^{23}d^2 + (1 - \delta^{23})d^3 + f = (0, 1)$ , and  $\delta^{31}d^3 + (1 - \delta^{31})d^1 + f = (0, 0)$ . Let  $\hat{u} = \left( \frac{1 - \delta^{12} + \delta^{23}}{2} \right) d^2 + (\delta^{12} + \delta^{31} - 1)d^1$ . Denote  $\mathbb{P}(r - \hat{u})$  by  $v_0$ . If  $V(v_0) = 1 - \pi(\hat{u})$ , then  $(\phi^{v_0}, \pi)$  is minimal for  $MI(I^2, S^2, r)$ . □*

We sketch the main steps in the proof of Theorem 6:

1. Construct a set  $T(\pi)$  defined as follows: (Refer to Fig. 5)
  - (a) Let  $Q^1$  be the quadrilateral whose vertices are:  $f + (1 - \delta^{12})d^2$  (represented by j),  $f + \delta^{23}d^2$  (represented by i),  $f + \delta^{23}d^2 + (\delta^{12} - 1 + \delta^{31})d^1$  (represented by o),  $f + (1 - \delta^{12})d^2 + (\delta^{12} - 1 + \delta^{31})d^1$  (represented by p).

- (b) Let  $Q^2$  be the quadrilateral whose vertices are:  $f + \delta^{31}d^3$  (represented by k),  $f + \delta^{31}d^3 + (\delta^{23} - 1 + \delta^{12})d^2$  (represented by q),  $f + (1 - \delta^{23})d^3 + (\delta^{23} - 1 + \delta^{12})d^2$  (represented by z),  $f + (1 - \delta^{23})d^3$  (represented by l).
- (c) Let  $Q^3$  be the quadrilateral whose vertices are:  $f + \delta^{12}d^1$  (represented by m),  $f + (1 - \delta^{31})d^1$  (represented by n),  $f + (1 - \delta^{31})d^1 + (\delta^{31} - 1 + \delta^{23})d^3$  (represented by t),  $f + \delta^{12}d^1 + (\delta^{31} - 1 + \delta^{23})d^3$  (represented by s).
- (d) Let  $T(\pi) = (D(\pi) + \{f\}) \cup Q^1 \cup Q^2 \cup Q^3$ . (Note  $D(\pi) + \{f\}$  is represented by *dihl*, *dken*, and *dmgj*).

2. Prove that  $\mathbb{P}(T(\pi)) = I^2$ . This involves applying a sequence of lattice-preserving operations to subsets of  $T(\pi)$  as shown in Fig. 6.
3. Refer to fig. 5. The point  $\hat{u} + f$  which is represented by  $u_0$ , is the center of the line segment  $op$ . Let  $u_{10}$  be the center of the line segment  $qk$ . Let  $\alpha$  be the center of  $ij$ , i.e.,  $\alpha = f + \frac{1}{2}(1 - \delta^{12} + \delta^{23})d^2$ . Let  $\beta$  be the center of  $zl$ , i.e.,  $\beta = f + (1 - \delta^{23})d^3 + \frac{1}{2}(1 - \delta^{12})d^2$ . We use the symbols  $Q^{11}$ ,  $Q^{12}$ ,  $Q^{21}$ , and  $Q^{22}$  to represent the quadrilateral  $\{\alpha, j, p, u_0\}$ ,  $\{\alpha, i, o, u_0\}$ ,  $\{k, l, \beta, u_{10}\}$ , and  $\{\beta, z, q, u_{10}\}$  respectively. Construct the function  $\phi_1 : T(\pi) \rightarrow \mathbb{R}_+$  as follows.

$$\phi_1(u) = \begin{cases} V_{v_0} + \pi(u - f - (1 - \delta^{31})d^1 + \frac{1}{2}(-1 + \delta^{12} + \delta^{23})d^2 - (0, 1)) & \text{if } u \in Q^{12} \\ V_{v_0} + \pi(u - f - (1 - \delta^{31})d^1 + \frac{1}{2}(-1 + \delta^{12} + \delta^{23})d^2 - (-1, 1)) & \text{if } u \in Q^{22} \\ V_{v_0} + \pi(u - f - (1 - \delta^{31})d^1 + \frac{1}{2}(-1 + \delta^{12} + \delta^{23})d^2) & \text{if } u \in Q^3 \\ \pi(u - f) & \text{otherwise} \end{cases}$$

4. Construct the function  $\tilde{\phi}(u) = \min\{\phi_1(w) | w \in T(\pi), \mathbb{P}(w - f) = u\}$ . Prove that  $\tilde{\phi} \geq \phi^{v_0}$ .
5. Prove that  $\tilde{\phi}(u) + \tilde{\phi}(r - u) = 1 \forall u \in I^2$ .

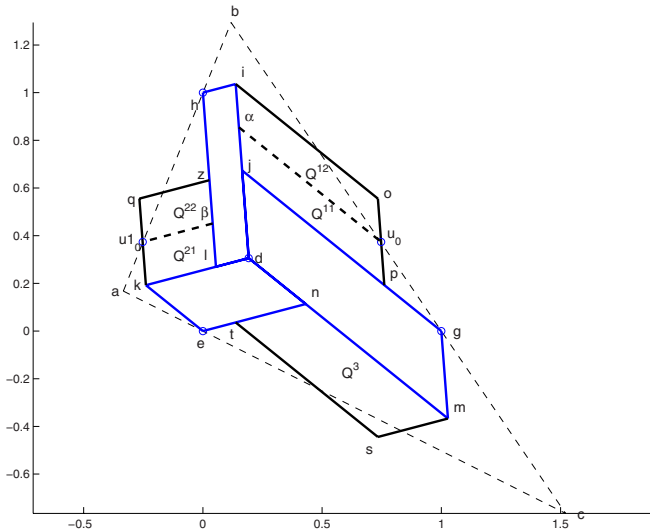


Fig. 5. Figure illustrating  $u_0$ ,  $\alpha$ , and  $\beta$

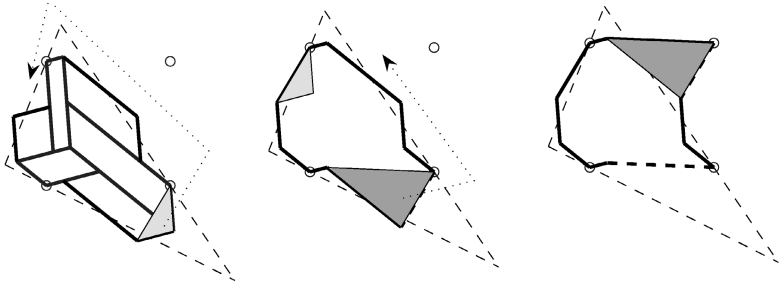


Fig. 6. Sequence of operations to prove that  $\mathbb{P}(T(\pi)) = I^2$

6. Now use the following theorem from Johnson [11]: If  $\phi : I^2 \rightarrow \mathbb{R}_+$  is a valid function for  $MI(I^2, \emptyset, r)$  and if  $\phi(u) + \phi(r - u) \leq 1 \forall u \in I^2$ , then  $\phi$  is subadditive. This shows that  $(\tilde{\phi}, \pi)$  is minimal and that  $\tilde{\phi} = \phi^{v_0}$ .

## 7 Conclusion

In this paper, we analyzed lifting functions for nonnegative integer variables when starting from minimal valid inequalities for a system of two rows with two free integer variables and nonnegative continuous variables. We proved that unique lifting functions exist in the case when the original inequality for the continuous variables corresponds to either a maximal lattice-free triangle with multiple integral points in the relative interior of one of its sides or a maximal lattice-free triangle with integral vertices and one integral point in the relative interior of each side. The resulting lifted inequality is minimal for  $MI(I^2, S^2, r)$ . In Theorem 6, we showed that under suitable conditions, starting with a specific cyclic subgroup of  $I^2$  and using the fill-in procedure leads to minimal inequalities for  $MI(I^2, S^2, r)$  when the inequality  $\pi$  corresponds to a lattice-free triangle with non-integral vertices and one integral point in the relative interior of each side.

We hope that these new families of minimal inequalities for  $MI(I^2, S^2, r)$  may perform well computationally, since the coefficient for continuous variables in these inequalities is not dominated by any other inequality for the two-dimensional infinite group problem. These inequalities may also prove to be a source for numerically stable high-rank cuts.

Future research directions include analysis of maximal lattice-free quadrilaterals and extensive computational experiments.

## References

[1] Andersen, K., Louveaux, Q., Weismantel, R., Wolsey, L.: Cutting planes from two rows of a simplex tableau. In: Fischetti, M., Williamson, D.P. (eds.) Proceedings 12<sup>th</sup> Conference on Integer and Combinatorial Optimization, pp. 30–42. Springer, Heidelberg (2007)



- [2] Balas, E.: Intersection cuts - a new type of cutting planes for integer programming. *Operations Research* 19, 19–39 (1971)
- [3] Borozan, V., Cornuéjols, G.: Minimal inequalities for integer constraints (2007), <http://integer.tepper.cmu.edu>
- [4] Cook, W.J., Kannan, R., Schrijver, A.: Chvátal closures for mixed integer programming problems. *Mathematical Programming* 58, 155–174 (1990)
- [5] Cornuéjols, G., Margot, F.: On the facets of mixed integer programs with two integer variables and two constraints (2007), [http://wpweb2.tepper.cmu.edu/fmargot/rec\\_pub.html](http://wpweb2.tepper.cmu.edu/fmargot/rec_pub.html)
- [6] Gomory, R.E.: Some polyhedra related to combinatorial problems. *Linear Algebra and Applications* 2, 341–375 (1969)
- [7] Gomory, R.E., Johnson, E.L.: Some continuous functions related to corner polyhedra, part I. *Mathematical Programming* 3, 23–85 (1972)
- [8] Gomory, R.E., Johnson, E.L.: Some continuous functions related to corner polyhedra, part II. *Mathematical Programming* 3, 359–389 (1972)
- [9] Gomory, R.E., Johnson, E.L.: T-space and cutting planes. *Mathematical Programming* 96, 341–375 (2003)
- [10] Gomory, R.E., Johnson, E.L., Evans, L.: Corner polyhedra and their connection with cutting planes. *Mathematical Programming* 96, 321–339 (2003)
- [11] Johnson, E.L.: On the group problem for mixed integer programming. *Mathematical Programming Study* 2, 137–179 (1974)
- [12] Nemhauser, G.L., Wolsey, L.A.: *Integer and Combinatorial Optimization*. Wiley-Interscience, New York (1988)

# Author Index

- Archer, Aaron 316  
Armbruster, Michael 112  
Balas, Egon 416  
Berger, André 273  
Bernáth, Attila 401  
Bertsimas, Dimitris 34  
Bonami, Pierre 17  
Bonifaci, Vincenzo 273  
Carnes, Tim 288  
Chakrabarty, Deeparnab 344  
Conforti, Michele 435  
Devanur, Nikhil R. 344  
Dey, Santanu S. 463  
Di Summa, Marco 435  
Espinoza, Daniel G. 214  
Fischetti, Matteo 416  
Fleiner, Tamás 385  
Fügenschuh, Marzena 112  
Georgiou, Konstantinos 140  
Grandoni, Fabrizio 273  
Guan, Yongpei 450  
Günlük, Oktay 1  
Halldórsson, Magnús M. 359  
Helmberg, Christoph 112  
Jansen, Klaus 184  
Kawarabayashi, Ken-ichi 47, 374  
Király, Tamás 259, 401  
Kobayashi, Yusuke 47  
Kortsarz, Guy 359  
Krishnan, Shankar 316  
Lau, Lap Chi 259  
Lee, Jon 17  
Letchford, Adam N. 125  
Linderoth, Jeff 1, 225  
Lulli, Guglielmo 34  
Magen, Avner 140  
Martens, Maren 97  
Martin, Alexander 112  
McCormick, S. Thomas 97  
Miller, Andrew 450  
Munagala, Kamesh 169  
Nagarajan, Chandrashekhar 303  
Nagarajan, Viswanath 154  
Nemhauser, George L. 199  
Odoni, Amedeo 34  
Oriolo, Gianpaolo 77  
Orlin, James B. 240  
Ostrowski, James 225  
Pietropaoli, Ugo 77  
Rossi, Fabrizio 225  
Saxena, Anureet 17  
Schäfer, Guido 273  
Shi, Peng 169  
Shmoys, David 288, 331  
Singh, Mohit 259  
Smriglio, Stefano 225  
Solis-Oba, Roberto 184  
Sørensen, Michael M. 125  
Stauffer, Gautier 77  
Sviridenko, Maxim 154, 359  
Takazawa, Kenjiro 62  
Talwar, Kunal 331  
Tourlakis, Iannis 140  
Vazirani, Vijay V. 344  
Vielma, Juan Pablo 199  
Williamson, David P. 303  
Wolsey, Laurence A. 435, 463  
Zanette, Arrigo 416