Michael Jünger
Volker Kaibel (Eds.)

LNCS 3509

Integer Programming and Combinatorial Optimization

11th International IPCO Conference
Berlin, Germany, June 2005
Proceedings

Springer

Lecture Notes in Computer Science 3509

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Michael Jünger Volker Kaibel (Eds.)

Integer Programming and Combinatorial Optimization

11th International IPCO Conference
Berlin, Germany, June 8-10, 2005
Proceedings

Springer

Volume Editors

Michael Jünger
Universität zu Köln
Institut für Informatik
Pohligstraße 1, 50969 Köln, Germany
E-mail: mjuenger@informatik.uni-koeln.de

Volker Kaibel
Zuse-Institut Berlin
Takustr. 7, 14195 Berlin, Germany
E-mail: kaibel@zib.de

Library of Congress Control Number: 2005926927

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

ISSN 0302-9743
ISBN-10 3-540-26199-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26199-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11496915 06/3142 5 4 3 2 1 0

Preface

Since its start in 1990, the IPCO conference series (held under the auspices of the Mathematical Programming Society) has become an important forum for the presentation of recent results in Integer Programming and Combinatorial Optimization. This volume compiles the papers presented at IPCO XI, the eleventh conference in this series, held June 8–10, 2005, at the Technische Universität Berlin.

The high interest in this conference series is evident in the large number of submissions. For IPCO XI, 119 extended abstracts of up to 10 pages were submitted. During its meeting on January 29–30, 2005, the Program Committee carefully selected 34 contributions for presentation in non-parallel sessions at the conference. The final choices were not easy at all, since, due to the limited number of time slots, many very good papers could not be accepted.

During the selection process the contributions were refereed according to the standards of refereed conferences. As a result of this procedure, you have in your hands a volume that contains papers describing high-quality research efforts. The page limit for contributions to this proceedings volume was set to 15. You may find full versions of the papers in scientific journals in the near future.

We thank all the authors who submitted papers. Furthermore, the Program Committee is indebted to the many reviewers who, with their specific expertise, helped a lot in making the decisions.

Aussois, March 2005

Michael Jünger
Volker Kaibel

Organization

IPCO XI was organized by the Technische Universität Berlin and by the DFG Research Center MATHEON under the auspices of the Mathematical Programming Society.

Program Committee

Michele Conforti (University of Padova)
Jesus De Loera (University of California, Davis)
Andras Frank (ELTE Budapest)
Dorit Hochbaum (University of California, Berkeley)
Michael Jünger, chair (Universität zu Köln)
Volker Kaibel (Zuse Institute Berlin)
Adam Letchford (Lancaster University)
Rolf Möhring (Technische Universität Berlin)
Petra Mutzel (Universität Dortmund)
Maurice Queyranne (University of British Columbia)
Franz Rendl (Universität Klagenfurt)
Bruce Shepherd (Bell Labs)

Organizing Committee

Claudia Ewel (Technische Universität Berlin)
Martin Grötschel (Technische Universität Berlin/Zuse Institute Berlin)
Volker Kaibel, chair (Zuse Institute Berlin)
Rafael Mechtel (Technische Universität Berlin)
Rolf Möhring (Technische Universität Berlin)
Matthias Peinhardt (Zuse Institute Berlin)
Marc Pfetsch (Zuse Institute Berlin)
Elke Pose (Technische Universität Berlin)
Sebastian Stiller (Technische Universität Berlin)

Industrial Sponsors

IBM

Lufthansa Systems

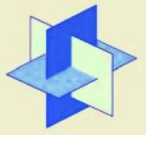
ILOG
Changing the rules of business™

CARMEN SYSTEMS
RESOURCES IN BALANCE

SIEMENS

Scientific Sponsors

DFG Research Center MATHEON



DFG Research Group *Algorithms, Structure, Randomness*

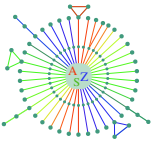


Table of Contents

Mixed-Integer Cuts from Cyclic Groups

Matteo Fischetti and Cristiano Saturni

DEI, Department of Information Engineering,University of Padua,
via Gradenigo 6/a, I-35100, Padova, Italy
{matteo.fischetti, cristiano.saturni}@dei.unipd.it
http://www.dei.unipd.it/

Abstract. We analyze a separation procedure for Mixed-Integer Programs related to the work of Gomory and Johnson on interpolated subadditive functions. This approach has its roots in the Gomory-Johnson characterization on the master cyclic group polyhedron. To our knowledge, the practical benefit that can be obtained by embedding interpolated subadditive cuts in a cutting plane algorithm was not investigated computationally by previous authors. In this paper we compute, for the first time, the lower bound value obtained when adding (implicitly) *all* the interpolated subadditive cuts that can be derived from the individual rows of an optimal LP tableau, thus approximating the optimization over the so-called Gomory's Corner polyhedron. The computed bound is compared with that obtained when only Gomory mixed-integer cuts are used, on a large test-bed of MIPLIB instances.

Keywords: Mixed-Integer Programming, Subadditive cuts, Gomory cuts, Gyclic Group and Corner polyhedra.

1 Introduction

In this paper we study the Integer Linear Program (ILP)

$$\min\{c^T x : Ax = b, x \geq 0 \text{ integer}\} \tag{1}$$

where A is a rational $m \times n$ matrix, b is a rational m-dimensional vector, and $c \in \mathbb{R}^n$ is a cost vector, and we address the two associated polyhedra:

$$P := \{x \in \mathbb{R}^n_+ : Ax = b\} \tag{2}$$

$$P_I := conv\{x \in \mathbb{Z}^n_+ : Ax = b\} = conv(P \cap \mathbb{Z}^n) \ . \tag{3}$$

The mixed-integer case where some variables are not restricted to be integer, will be addressed in Section 3.

We propose an exact separation procedure for the class of so-called *interpolated* (or *template*) *subadditive cuts*, based on the characterization of Gomory and Johnson [10, 11, 12] of the *master cyclic group polyhedron* defined as

$$T(k,r) = conv\{t \in \mathbb{Z}^{k-1}_+ : \sum_{i=1}^{k-1} (i/k) \cdot t_i \equiv r/k \ (\text{mod } 1)\} \tag{4}$$

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 1–11, 2005.

where $k \geq 2$ (group order) and $r \in \{1, \cdots, k-1\}$ are given integers. The space $\mathbb{R}^{k-1}$ of the t variables is called the *T-space* in [13]. It is known that the mapping the original x-variable space into the T-space allows one to use polyhedral information on $T(k,r)$ to derive valid inequalities for P_I. To our knowledge, however, the practical benefit that can be obtained by embedding these cuts in a cutting plane algorithm was not investigated computationally by previous authors. As a matter of fact, a number of recent papers [13, 14, 5, 6, 7, 2] deals only implicitly with cyclic-group separation, as they address the so-called Gomory's *shooting experiment*. Roughly speaking, in this experiment the point $t^* \in \mathbb{R}^{k-1}$ to be separated is generated at random (hence corresponding to a random "shooting direction" d in the T-space), and statistics on the frequency of the most-violated facets of $T(k,r)$ are collected. A very recent paper presenting some computational results is the one by Koppe, Louveaux, Weismantel and Wolsey [18], where a compact formulation of the cyclic-group separation problem is embedded into the original ILP model—this however produces a huge extended formulation with limited practical applications. Also related to our work are the papers by Cornuejols, Li and Vandenbussche [4], where a subfamily of cyclic-group cuts (called k-cuts) is investigated both theoretically and computationally, and by Letchford and Lodi [17], where a different subfamily is addressed.

This paper is organized as follows. In Section 2 we present the theory of Gomory and Johnson [10, 11, 12] on interpolated subadditive functions and their role in generating valid inequalities for P_I. We also introduce an exact separation procedure for interpolated subadditive cuts based on an LP model taken from the Gomory-Johnson characterization of the master cyclic group polyhedron. This separation procedure allows us to exploit effectively the whole family of interpolated subadditive cuts to improve the LP relaxation quality. In Section 3 we consider the mixed-integer case, where model (1) becomes a MIP involving continuous variables. In Section 4, the quality of the generated cuts is analyzed computationally. In particular we compute, for the first time, the lower bound value obtained when adding (implicitly) *all* the interpolated subadditive cuts that can be derived from the individual rows of an optimal LP tableau. This leads to an approximation[1] of the optimal value of the so-called Gomory's *Corner polyhedron* [12], hence giving a partial answer to the question posed in [14] (and also addressed in [18]) on the quality of this relaxation. The bound we compute is compared with that obtained when only Gomory mixed-integer cuts are used, on a large test-bed made by the instances of the MIPLIB [19] library. The outcome is that Gomory mixed-integer cuts play a very special role among subadditive cuts, in that they typically produce, alone, a lower bound increase which is comparable to that obtained when the whole family of cuts is considered. This result confirms the theoretical findings of Dash and Gunluk [6], who showed that interpolated subadditive cuts are dominated by Gomory mixed-integer cuts in

[1] Besides getting rid of the effects of interpolation, optimizing *exactly* over the corner polyhedron would require to take into account *all* the tableau rows simultaneously.

a probabilistic sense, as well as the computational experience of Cornuejols, Li and Vandenbussche [4] on the subfamily of k-cuts.

2 Cuts from Subadditive Functions

The *fractional part* $\phi(a)$ of a real value a is defined as

$$\phi(a) := a - \lfloor a \rfloor \ ,$$

where $\lfloor a \rfloor$ denotes the largest integer not greater than a. Given a positive integer k and two real values $a, b \in \mathbb{R}$, we write $a \equiv b \pmod{k}$ if $a - b$ is an integer multiple of k. In this paper we are interested in deriving valid inequalities for P_I that are not implied by the system $Ax = b$, $x \geq 0$. To this end, given any equation

$$\alpha^T x = \beta \tag{5}$$

valid for P_I, where $(\alpha, \beta) \in \mathbb{Q}^{n+1}$ and $\phi(\beta) > 0$, we consider the group polyhedron

$$G(\alpha, \beta) := conv\{x \in \mathbb{Z}^n_+ : \sum_{j=1}^{n} \alpha_j x_j \equiv \beta \pmod{1}\} \supseteq P_I \ . \tag{6}$$

For example, the equation $\alpha^T x = \beta$ can be obtained by setting $(\alpha, \beta)^T := u^T(A, b)$ for any $u \in \mathbb{Q}^m$ such that $\phi(u^T b) > 0$. This is the case, e.g., when the equation is read from the tableau associated with a fractional optimal solution of the LP relaxation of (1).

In particular, we address the following separation problem:

Definition 1. (g-SEP) *Given any point $x^* \geq 0$ and the equation $\alpha^T x = \beta$ with rational coefficients and such that $\phi(\beta) > 0$, find (if any) a valid inequality for $G(\alpha, \beta)$ that is violated by x^*.*

A function $g : \mathbb{R} \to \mathbb{R}$ is called subadditive if $g(a + b) \leq g(a) + g(b)$ for any $a, b \in \mathbb{R}$. We call a subadditive function $g(\cdot)$ periodic in $[0, 1)$ if $g(a + 1) = g(a)$ for all $a \in \mathbb{R}$. As in this paper we are only interested in subadditive functions $g(\cdot)$ that are periodic in $[0, 1)$ and such that $g(0) = 0$, in the sequel we will name this kind of functions just *subadditive*.

Given a valid equation $\alpha^T x = \beta$ for P_I, it is easy to show that the inequality

$$\sum_{j=1}^{n} g(\alpha_j) x_j \geq g(\beta) \tag{7}$$

is valid for $G(\alpha, \beta)$ (and hence for P_I) whenever $g(\cdot)$ is subadditive. For example, taking $g(\cdot) = \phi(\cdot)$ one obtains the well-know *Gomory fractional cut* [8]

$$\sum_{j=1}^{n} \phi(\alpha_j) x_j \geq \phi(\beta) \ ,$$

whereas taking the subadditive *GMI function* $\gamma^\beta(\cdot)$ defined as

$$\gamma^\beta(a) = \begin{cases} \phi(a) & \text{if } \phi(a) \le \phi(\beta) \\ \phi(\beta)\frac{1-\phi(a)}{1-\phi(\beta)} & \text{otherwise} \end{cases} \quad \text{for all } a \in \mathbb{R} \tag{8}$$

one obtains the stronger *Gomory Mixed-Integer* (GMI) cut [9]

$$\sum_{j=1}^{n} \min\{\phi(\alpha_j),\, \phi(\beta)\frac{1-\phi(\alpha_j)}{1-\phi(\beta)}\}\, x_j \ge \phi(\beta)\ . \tag{9}$$

A basic result, due to Gomory and Johnson [11, 12], is that *all* the nontrivial facets of $G(\alpha, \beta)$ are defined by inequalities of this type.[2] As a consequence, our separation problem (*g-SEP*) can be rephrased as the problem of defining a suitable subadditive function $g(\cdot)$ that produces a cut violated by the given point x^*, i.e., such that $\sum_{j=1}^{n} g(\alpha_j)x_j^* < g(\beta)$.

Now let $k \ge 2$ be the smallest integer such that $k(\alpha, \beta)$ is integer, whose existence follows from the assumption that (α, β) is rational. This value of k will be called *ideal* with respect to equation (5). Of course, the subadditivity (plus periodicity) of $g(\cdot)$ implies that the same property holds over the discrete set $\{0, 1/k, 2/k, \cdots, (k-1)/k\}$. In other words, a necessary condition for subadditivity is that the "sampled" values $g_i := g(i/k)$, $i = 0, \cdots, k-1$ satisfy the following set of linear constraints, called *g-system* in the sequel:

$$g_i + g_j \ge g_h\,, \quad 1 \le i, j, h \le k-1 \text{ and } i + j \equiv h \pmod{k} \tag{10}$$

$$g_0 = 0 \tag{11}$$

$$0 \le g_i \le 1\,, \quad i = 1, \cdots, k-1\ , \tag{12}$$

where bounds (12) will play a normalization role in the sequel.

Any solution $(g_0, \cdots, g_{k-1})$ of the *g*-system above can be completed so as to define a subadditive function $g : \mathbb{R} \to \mathbb{R}$ through a simple *interpolation procedure* due to Gomory and Johnson [11]. This procedure simply takes a linear interpolation of the values $g_0, \cdots, g_{k-1}$ over $[0, 1)$, and then extends the resulting piecewise-linear function to $\mathbb{R}$, in the obvious periodic way. More formally, for any $a \in \mathbb{R}$ the interpolated value $g(a)$ is defined as $g(a) = (1-\theta)g_i + \theta g_{i+1}$, where $\theta \in [0, 1)$ and $i \in \{0, \cdots, k-1\}$ are such that $\phi(a) = (1-\theta)\, i/k + \theta\,(i+1)/k$, and $g_k := g_0$ because of periodicity.

A key observation at this point is that, being k ideal, the actual value of $g(\cdot)$ outside the sample points i/k is immaterial, since $g(\cdot)$ only needs to be evaluated on these sample points when computing the coefficients in (7). Therefore, the interpolation procedure does not actually restrict the space of the possible subadditive functions—as it would be the case for a different choice of k. As a consequence, we can *exactly* rephrase *g-SEP* as the following LP:

[2] In addition, Gomory and Johnson have shown that non-dominated inequalities only arise when the complementarity condition $g(a) + g(b) = g(a+b) (= g(\ ))$ holds whenever $a + b \equiv \ \pmod 1$.

$$g - SEP_k : \quad \min\{ \sum_{i=1}^{k-1} t_i^* g_i : (10) - (12) \ \} \ , \tag{13}$$

where $r := k\,\phi(\beta)$, $t_i^* := \sum_j \{x_j^* : \phi(\alpha_j) = i/k\}$ for $i \in \{1, \cdots, k-1\} \setminus \{r\}$, and $t_r^* := \sum_j \{x_j^* : \phi(\alpha_j) = r/k\} - 1$ so as to take into account the role of the right hand side $g(\beta) = g_r$ in (7). With these definitions, the objective function $\sum_{i=1}^{k-1} t_i^* g_i$ is precisely the opposite of the violation of a cut of the form (7), hence a violated such cut exists if and only if the optimal value of $g - SEP_k$ is strictly negative.

Unfortunately, the ideal k is very often too large to be used in practice, so one has to choose a smaller value in order to produce a manageable g-system. In this case, the interpolation procedure does restrict (often considerably) the range of subadditive functions that can be captured by $g - SEP_k$. Moreover, the definition of the weights t_i^* becomes slightly more involved, due to the need of taking interpolation into account. More specifically, for any integer $k \geq 2$ (not necessarily ideal) the weights t_i^* in (13) are defined through the algorithm outlined in Figure 1. At step 1, we define two fictitious values α_0 and x_0^* so as to re-write the (opposite of the) cut violation $\sum_{j=1}^n g(\alpha_j)x_j^* - g(\beta)$ in the more convenient form $\sum_{j=0}^n g(\alpha_j)x_j^*$. At step 2, all weights t_i^* are initially set to zero. At step 3, for each $j = 0, \cdots, n$ we locate the interval $[i/k, (i+1)/k)$ that contains $\phi(\alpha_j)$, where $i+1$ is replaced by 0 in case $i + 1 = k$ so as to take periodicity into account. At step 4 we define a "displacement" parameter $\theta \in [0, 1)$ giving the exact position of $\phi(\alpha_j)$ within this interval; by definition, we have $\theta = 0$ if $\phi(\alpha_j) = i/k$, whereas θ approaches its limit 1 as $\phi(\alpha_j)$ approaches $(i+1)/k$. We then split, at step 5, the contribution $g(\alpha_j)\, x_j^*$ between t_i^* and t_{i+1}^*, in a way proportional to θ. Note that the procedure also works in case of ideal k, when we always have $\theta = 0$ at step 4.

```
1. define the fictitious values  _0 :=   and  *_0 := −1;
2. initialize  *_0 := *_1 := ··· := *_{k−1} := 0;
2. for j = 0, 1, ···,   such that  *_j > 0 and  (  _j) > 0 do
3.    let i := ⌊k  (  _j)⌋ and h = i + 1 mod k;
4.    let   := k  (  _j) − i;
5.    update  *_i := *_i + (1 −  ) *_j and  *_h := *_h +  *_j
6. enddo
```

Fig. 1. Defining the weights $^*_i$ in $g - SEP_k$ for any given k and *

We finally observe that, for the interpolated function $g(\cdot)$, we sometimes have $g(a) > g(\beta)$. Therefore, an interpolated subadditive cut $\sum_{j=1}^n g(\alpha_j)x_j \geq g(\beta)$ can easily be improved to its *clipped* form:

$$\sum_{j=1}^{n} \min\{g(\alpha_j), g(\beta)\}x_j \geq g(\beta) \tag{14}$$

whose validity follows trivially from the integrality of x.

3 Dealing with Continuous Variables

We next address the case where some variables x_j with $j \in \mathcal{C}$ (say) are not restricted to be integer valued. In this case, Gomory and Johnson [11, 12] showed that, for any subadditive function $g(\cdot)$, it is enough to modify cut (7) into

$$\sum_{j\in\mathcal{I}} g(\alpha_j)x_j + \sum_{j\in\mathcal{C}:\alpha_j>0} slope_+\,\alpha_j x_j + \sum_{j\in\mathcal{C}:\alpha_j<0} slope_-\,\alpha_j x_j \geq g(\beta)\ , \qquad (15)$$

where $\mathcal{I} := \{1, \cdots, n\} \setminus \mathcal{C}$ is the index set of the integer-valued variables,

$$slope_+ := \lim_{\delta\to 0^+} g(\delta)/\delta$$

is the slope of $g(\cdot)$ in 0^+, and

$$slope_- := \lim_{\delta\to 0^-} g(\delta)/\delta \ = -\lim_{\delta\to 0^+} g(1-\delta)/\delta$$

is the slope of $g(\cdot)$ in 0^- (or, equivalently, in 1^-). Notice that, by definition, $slope_+ > 0$ and $slope_- < 0$, hence all coefficients in (15) are nonnegative.

The above result has an intuitive explanation based on the following simple scaling argument. Let $j \in \mathcal{C}$ be the index of any continuous variable. We introduce a scaled copy $\tilde{x}_j = Mx_j$ of x_j, where $M > 0$ is a suitable scaling factor, and impose that $\tilde{x}_j$ can only assume integer values. This is of course correct only if M is chosen so as not to cut any feasible point of the original MIP set, which is always possible due the rational data assumption. A key observation is however that M can be assumed to be arbitrarily large, since multiplying a valid M by a positive integer yields another valid M. Now, replacing x_j by $\tilde{x}_j/M$ changes the j-th coefficient in equation $\alpha^T x = \beta$ from α_j to $\tilde{\alpha}_j := \alpha_j/M$, while increasing the j-th component of x^* from x^*_j to $\tilde{x}^*_j := Mx^*_j$. Being $\tilde{x}_j$ constrained to be integer, we can compute its coefficient in (7) as $g(\tilde{\alpha}_j) = g(\alpha_j/M)$. If $\alpha_j > 0$, for $M \to +\infty$ we have that α_j/M tends to 0^+, hence for sufficiently large M we have $g(\tilde{\alpha}_j) = g(\alpha_j/M) = slope_+\alpha_j/M$. Analogously, if $\alpha_j < 0$ we have $\alpha_j/M \to 0^-$ when $M \to +\infty$, hence for sufficiently large M we have $g(\tilde{\alpha}_j) = g(\alpha_j/M) = slope_-\alpha_j/M$. This shows that the coefficient of the scaled variable $\tilde{x}_j$ in the subadditive cut is $slope_*\,\alpha_j/M$, where $slope_* = slope_-$ or $slope_-$ depending on the sign of α_j. The back substitution $x_j = \tilde{x}_j/M$ then yields the coefficient $slope_*\,\alpha_j$ for the original (continuous) variable x_j, as in (15).

As a consequence of the above scaling argument, we can deal with continuous variables without any modification of our separation procedure, that can used as a black box. To this end, it is enough to implement a pre-processing scaling phase for the continuous variables, and then a post-processing phase where the separated cut returned by the black box is expressed in terms of the original (non-scaled) variables. In case of interpolated subadditive functions, a suitable scaling factor for each continuous variable x_j is $M_j := k|\alpha_j|$, that maps the original coefficient α_j into $\tilde{\alpha}_j = \alpha_j/M_j = \pm 1/k$. This shows that, in presence of

several continuous variables with nonzero $\alpha_j x_j^*$, values g_1 and g_{k-1} play a crucial role in the separation, the lower these values the better. Hence GMI cuts qualify as the strongest subadditive cuts when continuous variables are present, since they have the property of being associated with a subadditive function $\gamma^\beta(\cdot)$ where $slope_+$ and $slope_-$ are as small as possible; see [12].

Finally, we observe that the clipping of coefficient $g(\alpha_j)$ to $\min\{g(\alpha_j), g(\beta)\}$ in (14) is guaranteed to be valid only for integer-constrained variables x_j.

4 Computational Results

We next report a preliminary computational analysis aimed at comparing the quality of Gomory mixed-integer cuts with that of the interpolated sudadditive cuts, when embedded in a pure cutting plane method. For these preliminary experiments, our test-bed includes a sample of MIPLIB instances taken form [19] (more instances will be considered in the full paper). In order to avoid any dependency of the experiments from the way the slack variables and the variable bounds are dealt with, for the MIPs involving "$\leq$" or "$\geq$" constraints and/or bounds on the variables we built an equivalent MIP formulation in standard form, that includes the variable bound constraints and slack variables in an explicit way (though this choice penalizes computing times considerably). For the same reason, we did not consider the coefficient clipping (14). All LP's have been solved through the commercial software ILOG-Cplex 9.0 [15, 16].

Our order of business is to approximate the optimization over the Gomory's corner polyhedron associated with the optimal solution of the LP relaxation of our MIP model (without any MIP preprocessing). To this end, after the solution of first LP relaxation of our model, we store in our equation pool all the tableau rows $\alpha^T x = \beta$ with fractional right-hand side β. This pool is never updated during the run, i.e., we deliberately avoid generating subadditive cuts of rank greater than 1 (or, to be more precise, we avoid cuts derived from equations different from those associated with the single rows of the first LP tableau). The same applies to Gomory mixed-integer cuts, that have been derived from the equations in the pool and added (at once) to the LP. At each round of separation, at most 200 cuts are generated. Each run is aborted at the root node, i.e., no branching is allowed.

Table 1 and Table 2 give the outcome of our experiments. In the first column of the tables we report the problem name along with the computing time and the value L_c of the LP relaxation of the initial model. The remaining columns correspond to Gomory mixed-integer (GMI) and interpolated subadditive cuts for various values of the group-order parameter k. For each problem we report the final lower bound value L (and, in parenthesis, the ratio $\frac{L-L_c}{L_{GMI}-L_c}$, where L_{GMI} is the final Gomory mixed-integer lower bound), the total computing time T (and, in parenthesis, the ratio $\frac{T}{T_{GMI}}$, where T_{GMI} is the total computing time for the GMI cuts), the separation time S (and, in parenthesis, the ratio $\frac{S}{T}$) and the total number of generated cuts. The entries marked by a star correspond to

Table 1. Computational results (Part I)

Problem	GMI	K = 10	K = 20	K = 30	K = 60
10teams 0.170 917.000	921.00 (1.0) 3.47 (1.0) 0.44 (0.1) 200	921.00 (1.0) 4.52 (1.3) 1.30 (0.3) 200	921.00 (1.0) 4.76 (1.4) 1.58 (0.3) 200	921.00 (1.0) 5.43 (1.6) 2.30 (0.4) 200	921.00 (1.0) 15.71 (4.5) 12.43 (0.8) 200
a1c1s1 0.180 997.530	2996.27 (1.0) 1.16 (1.0) 0.21 (0.2) 200	2639.39 (0.8) 6.69 (5.8) 5.57 (0.8) 200	2795.01 (0.9) 15.49 (13.3) 13.32 (0.9) 356	2851.22 (0.9) 18.17 (15.6) 15.58 (0.9) 400	2932.56 (1.0) 69.07 (59.5) 66.65 (1.0) 442
aflow30a 0.020 983.167	1003.81 (1.0) 0.08 (1.0) 0.02 (0.3) 62	1001.97 (0.9) 0.42 (5.3) 0.25 (0.6) 106	1002.93 (1.0) 0.66 (8.3) 0.49 (0.7) 110	1003.19 (1.0) 1.87 (23.4) 1.60 (0.9) 132	1003.47 (1.0) 17.83 (222.5) 17.61 (1.0) 132
aflow40b 0.100 1005.665	1014.31 (1.0) 0.41 (1.0) 0.05 (0.1) 76	1014.03 (1.0) 2.64 (6.4) 1.69 (0.6) 114	1014.23 (1.0) 3.78 (9.2) 2.62 (0.7) 128	1014.28 (1.0) 4.73 (11.5) 3.65 (0.8) 132	1014.30 (1.0) 27.00 (65.8) 26.03 (1.0) 122
air03 0.591 338864.250	340160.00 (1.0) 2.23 (1.0) 0.29 (0.1) 72	340160.00 (1.0) 2.73 (1.2) 0.86 (0.3) 72	340160.00 (1.0) 2.63 (1.2) 0.85 (0.3) 72	340160.00 (1.0) 2.96 (1.3) 0.96 (0.3) 72	340160.00 (1.0) 4.36 (2.0) 2.54 (0.6) 72
air04 38.686 55535.436	55581.69 (1.0) 216.40 (1.0) 2.75 (0.0) 200	55579.31 (0.9) 331.88 (1.5) 13.38 (0.0) 266	55581.64 (1.0) 387.93 (1.8) 19.20 (0.0) 318	55583.69 (1.0)⋆ 521.54 (2.4) 20.82 (0.0) 374	55584.80 (1.1)⋆ 844.80 (3.9) 293.90 (0.3) 404
air05 3.024 25877.609	25900.66 (1.0) 34.46 (1.0) 3.33 (0.1) 202	25900.45 (1.0) 70.62 (2.0) 13.96 (0.2) 270	25901.92 (1.1)⋆ 63.31 (1.8) 13.40 (0.2) 286	25902.29 (1.1)⋆ 112.90 (3.3) 23.80 (0.2) 384	25903.37 (1.1)⋆ 355.42 (10.3) 274.21 (0.8) 472
bell3a 0.010 862578.643	872157.51 (1.0) 0.02 (1.0) 0.00 (0.0) 82	872015.80 (1.0) 0.12 (6.0) 0.10 (0.8) 143	872092.20 (1.0) 0.31 (15.5) 0.27 (0.9) 154	872099.44 (1.0) 0.68 (34.0) 0.64 (0.9) 150	872140.08 (1.0) 10.04 (501.5) 10.02 (1.0) 149
bell5 0.010 8608417.947	8660422.46 (1.0) 0.01 (1.0) 0.00 (0.0) 64	8654669.96 (0.9) 0.08 (8.0) 0.06 (0.7) 96	8657274.83 (0.9) 0.24 (24.0) 0.22 (0.9) 109	8658662.54 (1.0) 0.72 (72.0) 0.70 (1.0) 107	8661152.43 (1.0)⋆ 16.12 (1610.1) 16.09 (1.0) 129
cap6000 0.310 −2451537.325	−2451470.55 (1.0) 0.69 (1.0) 0.07 (0.1) 11	−2451474.73 (0.9) 1.04 (1.5) 0.19 (0.2) 14	−2451474.73 (0.9) 1.37 (2.0) 0.32 (0.2) 20	−2451472.58 (1.0) 1.12 (1.6) 0.29 (0.3) 14	−2451472.49 (1.0) 2.90 (4.2) 2.07 (0.7) 14
dano3mip 147.682 576.232	576.39 (1.0) 211.71 (1.0) 6.36 (0.0) 200	576.38 (1.0) 280.02 (1.3) 55.95 (0.2) 224	576.38 (1.0) 384.41 (1.8) 76.73 (0.2) 238	576.38 (1.0) 353.49 (1.7) 85.82 (0.2) 248	576.39 (1.0) 443.55 (2.1) 183.23 (0.4) 258
fiber 0.020 156082.518	316631.18 (1.0) 0.12 (1.0) 0.00 (0.0) 84	262906.07 (0.7) 0.53 (4.4) 0.35 (0.7) 148	311294.39 (1.0) 1.17 (9.7) 0.93 (0.8) 190	316718.91 (1.0)⋆ 2.95 (24.6) 2.61 (0.9) 214	318754.56 (1.0)⋆ 41.32 (343.8) 41.00 (1.0) 194
flugpl 0.000 1167185.726	1171213.72 (1.0) 0.00 (−) 0.00 (−) 25	1171004.98 (0.9) 0.02 (−) 0.02 (1.0) 47	1171142.46 (1.0) 0.07 (−) 0.07 (1.0) 52	1171193.55 (1.0) 0.28 (−) 0.27 (1.0) 48	1171197.64 (1.0) 5.08 (−) 5.08 (1.0) 50
gen 0.050 112130.041	112243.01 (1.0) 0.12 (1.0) 0.02 (0.2) 87	112221.20 (0.8) 1.54 (12.8) 1.31 (0.9) 129	112226.91 (0.9) 1.84 (15.3) 1.60 (0.9) 149	112233.89 (0.9) 3.56 (29.6) 3.26 (0.9) 151	112244.20 (1.0)⋆ 49.32 (410.4) 48.95 (1.0) 168
gesa2-o 0.080 25476489.678	25568096.68 (1.0) 0.32 (1.0) 0.06 (0.2) 194	25565904.71 (1.0) 2.87 (9.0) 2.31 (0.8) 358	25566983.51 (1.0) 4.97 (15.5) 4.33 (0.9) 400	25567437.57 (1.0) 8.66 (27.0) 7.90 (0.9) 381	25567679.73 (1.0) 87.19 (272.1) 86.40 (1.0) 371
gesa2 0.080 25476489.678	25568096.68 (1.0) 0.32 (1.0) 0.06 (0.2) 194	25565904.71 (1.0) 2.87 (9.0) 2.31 (0.8) 358	25566983.51 (1.0) 4.97 (15.5) 4.33 (0.9) 400	25567437.57 (1.0) 8.66 (27.0) 7.90 (0.9) 381	25567679.73 (1.0) 87.19 (272.1) 86.40 (1.0) 371
gesa2_o 0.100 25476489.678	25568096.68 (1.0) 0.32 (1.0) 0.07 (0.2) 194	25565904.71 (1.0) 3.02 (9.4) 2.50 (0.8) 358	25566983.51 (1.0) 5.34 (16.7) 4.55 (0.9) 400	25567437.57 (1.0) 8.62 (26.9) 7.79 (0.9) 381	25567679.73 (1.0) 88.65 (276.6) 87.92 (1.0) 371
gesa3 0.110 27833632.451	27905722.41 (1.0) 0.40 (1.0) 0.05 (0.1) 185	27903677.78 (1.0) 4.36 (10.9) 3.77 (0.9) 276	27904785.63 (1.0) 7.53 (18.8) 6.70 (0.9) 324	27905660.84 (1.0) 12.12 (30.3) 11.18 (0.9) 354	27905671.58 (1.0) 120.32 (300.4) 119.27 (1.0) 324
gesa3_o 0.090 27833632.451	27930183.75 (1.0) 0.34 (1.0) 0.06 (0.2) 200	27927124.73 (1.0) 3.19 (9.4) 2.57 (0.8) 325	27929027.47 (1.0) 5.74 (16.9) 4.81 (0.8) 418	27929810.37 (1.0) 10.14 (29.8) 9.20 (0.9) 414	27929902.68 (1.0) 91.47 (268.6) 90.61 (1.0) 389
l152lav 0.080 4656.364	4664.09 (1.0) 0.58 (1.0) 0.13 (0.2) 98	4663.89 (1.0) 1.98 (3.4) 0.75 (0.4) 176	4664.53 (1.1)⋆ 3.49 (6.0) 1.53 (0.4) 310	4665.04 (1.1)⋆ 7.05 (12.1) 3.92 (0.6) 384	4665.51 (1.2)⋆ 103.89 (178.9) 99.02 (1.0) 450
liu 0.070 346.000	560.00 (1.0) 0.81 (1.0) 0.19 (0.2) 316	560.00 (1.0) 6.81 (8.4) 5.48 (0.8) 566	560.00 (1.0) 10.33 (12.7) 8.49 (0.8) 650	560.00 (1.0) 12.62 (15.6) 10.27 (0.8) 604	560.00 (1.0) 51.55 (63.6) 49.45 (1.0) 614
lseu 0.000 834.682	945.62 (1.0) 0.01 (1.0) 0.00 (0.0) 23	945.55 (1.0) 0.05 (5.0) 0.03 (0.6) 39	946.27 (1.0)⋆ 0.11 (11.0) 0.10 (0.9) 43	947.96 (1.0)⋆ 0.55 (55.0) 0.54 (1.0) 47	948.32 (1.0)⋆ 13.37 (1335.1) 13.36 (1.0) 51

Table 2. Computational results (Part II)

Problem	GMI	K = 10	K = 20	K = 30	K = 60
manna81 0.200 −13297.000	−13164.00 (1.0) 7.22 (1.0) 2.45 (0.3) 800	−13164.00 (1.0) 29.05 (4.0) 22.69 (0.8) 800	−13164.00 (1.0) 29.64 (4.1) 24.16 (0.8) 800	−13164.00 (1.0) 31.11 (4.3) 25.02 (0.8) 800	−13164.00 (1.0) 37.80 (5.2) 32.11 (0.8) 800
mas74 0.000 10482.795	10570.72 (1.0) 0.02 (1.0) 0.00 (0.0) 24	10570.56 (1.0) 0.11 (5.5) 0.05 (0.5) 62	10576.53 (1.1)⋆ 0.39 (19.5) 0.30 (0.8) 96	10581.31 (1.1)⋆ 1.51 (75.5) 1.39 (0.9) 122	10585.77 (1.2)⋆ 47.16 (2354.5) 46.95 (1.0) 148
mas76 0.000 38893.904	38965.29 (1.0) 0.02 (1.0) 0.01 (0.5) 22	38968.33 (1.0)⋆ 0.05 (2.5) 0.03 (0.6) 50	38972.76 (1.1)⋆ 0.34 (17.0) 0.26 (0.8) 88	38975.64 (1.1)⋆ 1.07 (53.5) 1.00 (0.9) 80	38977.75 (1.2)⋆ 31.67 (1581.0) 31.59 (1.0) 108
mitre 0.531 114740.518	115082.83 (1.0) 15.65 (1.0) 4.85 (0.3) 800	115067.80 (1.0) 89.03 (5.7) 72.11 (0.8) 925	115080.07 (1.0) 91.96 (5.9) 75.59 (0.8) 912	115082.80 (1.0) 292.09 (18.7) 254.60 (0.9) 1029	115082.79 (1.0) 868.26 (55.5) 831.83 (1.0) 1003
mkc 0.160 −611.850	−608.84 (1.0) 1.16 (1.0) 0.22 (0.2) 177	−609.53 (0.8) 10.13 (8.7) 7.42 (0.7) 393	−608.00 (1.3)⋆ 16.59 (14.3) 12.38 (0.7) 588	−606.52 (1.8)⋆ 26.83 (23.1) 21.17 (0.8) 743	−606.20 (1.9)⋆ 206.67 (177.9) 201.43 (1.0) 770
mod010 0.130 6532.083	6535.50 (1.0) 0.58 (1.0) 0.12 (0.2) 68	6535.46 (1.0) 1.12 (1.9) 0.43 (0.4) 72	6535.46 (1.0) 1.42 (2.4) 0.51 (0.4) 76	6535.75 (1.1)⋆ 1.55 (2.7) 0.72 (0.5) 76	6536.00 (1.1)⋆ 9.63 (16.6) 8.45 (0.9) 92
opt1217 0.020 −20.021	−19.13 (1.0) 0.12 (1.0) 0.02 (0.2) 58	−19.17 (1.0) 0.54 (4.5) 0.26 (0.5) 120	−19.16 (1.0) 0.92 (7.7) 0.60 (0.7) 122	−19.14 (1.0) 2.08 (17.3) 1.61 (0.8) 130	−19.14 (1.0) 32.30 (268.7) 31.79 (1.0) 166
p0033 0.000 2520.572	2843.55 (1.0) 0.00 (−) 0.00 (−) 13	2641.20 (0.4) 0.02 (−) 0.01 (0.5) 21	2700.41 (0.6) 0.06 (−) 0.06 (1.0) 26	2756.17 (0.7) 0.38 (−) 0.38 (1.0) 34	2847.69 (1.0)⋆ 13.68 (−) 13.68 (1.0) 39
p0201 0.010 6875.000	7030.56 (1.0) 0.02 (1.0) 0.00 (0.0) 42	7030.56 (1.0) 0.12 (6.0) 0.03 (0.2) 104	7030.56 (1.0) 0.12 (6.0) 0.06 (0.5) 104	7030.56 (1.0) 0.12 (6.0) 0.07 (0.6) 84	7030.56 (1.0) 1.71 (85.5) 1.64 (1.0) 104
p0282 0.010 176867.503	179882.58 (1.0) 0.03 (1.0) 0.00 (0.0) 58	179711.49 (0.9) 0.15 (5.0) 0.09 (0.6) 79	179830.00 (1.0) 0.30 (10.0) 0.26 (0.9) 79	179784.53 (1.0) 0.80 (26.7) 0.76 (0.9) 81	179830.00 (1.0) 15.42 (513.3) 15.36 (1.0) 85
p0548 0.020 315.255	3567.50 (1.0) 0.07 (1.0) 0.00 (0.0) 98	684.07 (0.1) 0.30 (4.3) 0.20 (0.7) 142	3089.74 (0.9) 0.45 (6.4) 0.36 (0.8) 140	3004.52 (0.8) 1.26 (18.0) 1.13 (0.9) 152	3358.15 (0.9) 30.32 (432.6) 30.15 (1.0) 190
p2756 0.080 2688.750	2690.75 (1.0) 0.26 (1.0) 0.01 (0.0) 81	2690.73 (1.0) 1.42 (5.5) 0.90 (0.6) 138	2690.93 (1.1)⋆ 1.88 (7.2) 1.34 (0.7) 150	2690.93 (1.1)⋆ 3.99 (15.3) 3.19 (0.8) 175	2691.06 (1.2)⋆ 73.66 (282.9) 72.71 (1.0) 178
pp08a 0.000 2748.345	5157.91 (1.0) 0.03 (1.0) 0.01 (0.3) 104	5065.83 (1.0) 0.15 (5.0) 0.12 (0.8) 174	5122.51 (1.0) 0.43 (14.3) 0.38 (0.9) 188	5127.46 (1.0) 1.17 (39.0) 1.12 (1.0) 212	5151.88 (1.0) 23.96 (797.7) 23.90 (1.0) 206
pp08aCUTS 0.020 5480.606	6049.59 (1.0) 0.04 (1.0) 0.00 (0.0) 92	6023.68 (1.0) 0.35 (8.7) 0.27 (0.8) 136	6042.14 (1.0) 0.68 (17.0) 0.58 (0.9) 156	6041.07 (1.0) 1.86 (46.5) 1.76 (0.9) 150	6046.72 (1.0) 26.24 (655.0) 26.15 (1.0) 174
protfold 2.654 −41.957	−41.54 (1.0) 10.12 (1.0) 1.25 (0.1) 200	−41.55 (1.0) 27.50 (2.7) 11.17 (0.4) 600	−41.46 (1.2)⋆ 42.06 (4.2) 16.25 (0.4) 800	−41.44 (1.2)⋆ 69.14 (6.8) 33.63 (0.5) 1400	−41.38 (1.4)⋆ 173.81 (17.2) 127.40 (0.7) 2000
qiu 0.310 −931.639	−923.65 (1.0) 0.61 (1.0) 0.06 (0.1) 74	−924.23 (0.9) 2.62 (4.3) 1.67 (0.6) 98	−923.92 (1.0) 2.89 (4.7) 2.12 (0.7) 86	−923.81 (1.0) 5.38 (8.8) 4.04 (0.8) 98	−923.81 (1.0) 38.37 (62.8) 36.89 (1.0) 96
set1ch 0.010 32007.730	40594.41 (1.0) 0.11 (1.0) 0.01 (0.1) 200	38761.54 (0.8) 0.82 (7.5) 0.64 (0.8) 412	40050.02 (0.9) 1.59 (14.5) 1.33 (0.8) 418	40393.42 (1.0) 2.77 (25.2) 2.54 (0.9) 420	40543.87 (1.0) 42.74 (388.0) 42.49 (1.0) 438
seymour 6.429 403.846	405.47 (1.0) 18.72 (1.0) 3.71 (0.2) 237	405.46 (1.0) 51.33 (2.7) 28.97 (0.6) 360	405.47 (1.0) 53.24 (2.8) 34.38 (0.6) 349	405.47 (1.0) 63.34 (3.4) 44.95 (0.7) 360	405.47 (1.0) 400.21 (21.4) 373.95 (0.9) 436
timtab1 0.010 28694.000	202346.60 (1.0) 0.07 (1.0) 0.03 (0.4) 200	200669.04 (1.0) 0.56 (8.0) 0.46 (0.8) 316	201681.99 (1.0) 1.41 (20.1) 1.23 (0.9) 362	201783.03 (1.0) 3.88 (55.3) 3.67 (0.9) 398	202346.60 (1.0) 56.60 (807.4) 56.31 (1.0) 418
timtab2 0.020 83592.000	263570.66 (1.0) 0.17 (1.0) 0.01 (0.1) 234	259881.45 (1.0) 1.87 (11.0) 1.54 (0.8) 442	262459.44 (1.0) 2.83 (16.6) 2.46 (0.9) 468	263007.56 (1.0) 6.23 (36.6) 5.75 (0.9) 486	263570.66 (1.0) 93.90 (551.5) 93.44 (1.0) 484
vpm1 0.000 15.417	15.88 (1.0) 0.02 (1.0) 0.01 (0.5) 32	15.81 (0.9) 0.18 (9.0) 0.15 (0.8) 50	15.86 (1.0) 0.31 (15.5) 0.27 (0.9) 52	15.86 (1.0) 0.90 (45.0) 0.86 (1.0) 54	15.88 (1.0) 7.73 (386.0) 7.70 (1.0) 46
vpm2 0.000 9.889	10.30 (1.0) 0.04 (1.0) 0.00 (0.0) 62	10.29 (1.0) 0.31 (7.7) 0.27 (0.9) 82	10.29 (1.0) 0.61 (15.2) 0.55 (0.9) 106	10.29 (1.0) 1.95 (48.7) 1.82 (0.9) 106	10.30 (1.0) 31.36 (782.7) 31.30 (1.0) 118

the cases where our cutting plane method produced a better bound than GMI. All computing times are expressed in CPU seconds on a PC Athlon 1500.

For example, according to the first column of Table 1, the first problem in our test-bed is *10teams*, whose LP relaxation takes 0.170 seconds and produces a lower bound of 917.000. Our cutting plane method with $k = 60$ (last column in the table) produces a lower bound of 921.00 (the same as GMI) after 15.71 seconds (4.5 times the time required by GMI), with 12.43 seconds spent within our separation procedure (0.8 times the overall computing time) to generate 200 cuts.

The tables show that interpolated subadditive cuts typically become competitive with (or better than) GMI cuts for $k \geq 20$, though their separation requires a substantial computing-time overhead. Moreover, the large number of subadditive cuts generated and the small improvement obtained in some cases, would suggest a more conservative policy that generates GMI cuts first, and only afterward resorts to *g-SEP* to generate new violated subadditive cuts derived from the pool equations (if any). To be more specific, we believe that a better compromise between lower bound quality and computing time could be reached if one uses first a clever set of non-interpolated subadditive functions to derive quickly an initial set of violated inequalities (including GMI and k-cuts), and applies *g-SEP* separation only afterwards. This goes into the direction suggested by Andreello, Caprara and Fischetti [1] for an effective use of easy-to-compute cuts such as GMI and k-cuts.

References

1. G. Andreello, A. Caprara, M. Fischetti, Embedding Cuts in a Branch and Cut Framework: a Computational Study with $\{0, 1/2\}$-Cuts, Technical Report, University of Padova (2003); available at `www.dei.unipd.it/~fisch/papers`.
2. J. Araoz, R.E. Gomory, E.L. Johnson, and L. Evans, Cyclic group and knapsack facets, Mathematical Programming 96, 377-408 (2003).
3. E. Balas, S. Ceria, G. Cornuejols, N. Natraj, Gomory Cuts Revisited, Operations Research Letters 19, 1-9 (1996).
4. G. Cornuejols, Y. Li and D. Vandenbussche, K-Cuts: A Variation of Gomory Mixed Integer Cuts from the LP Tableau, INFORMS Journal on Computing 15, 385-396 (2003).
5. S. Dash and O. Gunluk, Valid inequalities based on simple mixed-integer sets, IBM Research Report RC22922, T. J. Watson Research Center, Yorktown Heights, New York, (2003).
6. S. Dash and O. Gunluk, Valid inequalities based on the interpolation procedure, IBM Research Report RC22922, T. J. Watson Research Center, Yorktown Heights, New York, (2004).
7. L. Evans, Cyclic Groups and Knapsack Facets with Applications to Cutting Planes, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, Georgia, (2002).
8. R.E. Gomory, Outline of an algorithm for integer solutions to linear programs, Bulletin of the AMS 64, 275-278 (1958).
9. R.E. Gomory, An algorithm for the mixed-integer problem. Report RM-2597, Rand Corporation (1960).

10. R.E. Gomory, Some Polyhedra Related to Combinatorial Problems, Journal of Linear Algebra and its Applications 2, 451-558 (1969).
11. R.E. Gomory and E.L. Johnson, Some Continuous Functions Related to Corner Polyhedra I, Mathematical Programming 3, 23-85 (1972).
12. R.E. Gomory and E.L. Johnson, Some Continuous Functions Related to Corner Polyhedra II, Mathematical Programming 3, 359-389 (1972).
13. R.E. Gomory and E.L. Johnson, T-space and cutting planes, Mathematical Programming 96, 341-375 (2003).
14. R.E. Gomory, E. Johnson, and L. Evans, Corner Polyhedra and their connection with cutting planes, Mathematical Programming 96, 321-339 (2003).
15. ILOG Cplex 9.0: User's Manual and Reference Manual, ILOG, S.A., `http://www.ilog.com/` (2004).
16. ILOG Concert Technology 2.0: User's Manual and Reference Manual, ILOG, S.A., `http://www.ilog.com/` (2004).
17. Letchford A.N. and A. Lodi, Strengthening Chvàtal-Gomory cuts and Gomory fractional cuts, Operations Research Letters 30(2), 74-82, 2002.
18. M. Koppe, Q. Louveaux, R. Weismantel and L.A. Wolsey, Extended Formulations for Gomory Corner Polyhedra, Discrete Optimization Discrete Optimization 1, 141-165, 2004.
19. MIPLIB - Mixed Integer Problem Library 2003, `http://miplib.zib.de` (2003)
20. G.L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization, Wiley, New York (1988).

Optimizing over the First Chvàtal Closure

Matteo Fischetti[1] and Andrea Lodi[2,3]

[1] DEI, University of Padova, via Gradenigo,
6A - 35131 Padova - Italy
matteo.fischetti@unipd.it
[2] DEIS, University of Bologna, viale Risorgimento,
2 - 40136 Bologna - Italy
[3] IBM T.J. Watson Research Center, P.O. Box 218,
Yorktown Heights, NY 10598
alodi@us.ibm.com

Abstract. How difficult is, in practice, to optimize *exactly* over the first Chvàtal closure of a generic ILP? Which fraction of the integrality gap can be closed this way, e.g., for some hard problems in the MIPLIB library? Does it pay to insist on rank-1 Chvàtal-Gomory inequalities until no such inequality is violated, or one should better follow the usual strategy of generating (mixed-integer) Gomory cuts of any rank from the optimal tableau rows? How effective is this general-purpose approach for solving matching problems, where the first Chvàtal closure coincides with the integer hull? Can this approach be useful as a research (off-line) tool to guess the structure of some relevant classes of inequalities, when a specific combinatorial problem is addressed? In this paper we give, for the first time, concrete answers to the above questions, based on an extensive computational analysis. Our approach is to model the rank-1 Chvàtal-Gomory separation problem, which is known to be NP-hard, through a MIP model, which is then solved through a general-purpose MIP solver. As far as we know, this approach was never implemented and evaluated computationally by previous authors, though it gives a very useful separation tool for general ILP problems. We report the optimal value over the first Chvàtal closure for a set of ILP problems from MIPLIB 3.0. We also report, for the first time, the optimal solution of a very hard instance from MIPIB 2003, namely *nsrand-ipx*, obtained by using our cut separation procedure to preprocess the original ILP model. Finally, we describe a new class of ATSP facets found with the help of our separation procedure.

Keywords: Integer Programs, separation problems, Chvàtal-Gomory cuts, computational analysis.

1 Introduction

We consider the Integer Linear Program (ILP) problem

$$\min\{c^T x : Ax \leq b, x \geq 0 \text{ integer}\} \tag{1}$$

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 12–22, 2005.

where A is a $m \times n$ matrix, $b \in I\!R^m$, and $c \in I\!R^n$, along with the two associated polyhedra:

$$P := \{x \in I\!R^n_+ : Ax \leq b\} \tag{2}$$
$$P_I := conv\{x \in Z^n_+ : Ax \leq b\} = conv(P \cap Z^n) \tag{3}$$

Unless explicitly stated, we assume that (A, b) is an integer matrix. Moreover, we assume that P_I is strictly contained in P, i.e., that P has fractional vertices.

A *Chvàtal-Gomory* (*CG*) *cut* [11, 7] is a valid inequality for P_I of the form

$$\lfloor u^T A \rfloor x \leq \lfloor u^T b \rfloor \,, \tag{4}$$

where $u \in R^m_+$ is called the *CG multiplier vector*, and $\lfloor \cdot \rfloor$ denotes lower integer part. Note that CG cuts depend on P and not directly on P_I, i.e., different formulations $Ax \leq b, x \geq 0$ of the same integer problem can produce different CG cuts. Inequality (4) is said of *rank 1* with respect to formulation $Ax \leq b$, $x \geq 0$; CG cuts of rank $h \geq 2$ are obtained in a similar way, starting with an enlarged system containing all CG cuts of rank less than h. The first *Chvàtal closure* of P is defined as

$$P_1 := \{x \geq 0 : Ax \leq b, \lfloor u^T A \rfloor x \leq \lfloor u^T b \rfloor \text{ for all } u \in I\!R^m_+\} \tag{5}$$

A basic result, due to Chvàtal [7], is that P_1 is indeed a polyhedron, i.e., a finite number of CG cuts suffice to define it.

Clearly, $P_I \subseteq P_1 \subseteq P$. It was shown by Gomory [11, 12] that every fractional vertex x^* of P associated with a certain basis B (say) of (A, I) can be cut off by the CG cut in which u is chosen as the i-th row of B^{-1}, where i is the row associated with any fractional component of x^*. Therefore, $P_1 \subset P$ in case $P \neq P_I$, i.e., P_1 gives a better approximation of P_I than P. In some cases, one has that $P_I = P_1$ as, e.g., for matching problems where undominated CG cuts correspond to the famous Edmonds' blossom inequalities [8, 15]. Even in case $P_I \subset P_1$, however, we are interested in optimizing $c^T x$ over P_1 in order to get a hopefully tight lower bound on the optimal value of the original ILP problem (1).

By the well-known equivalence between optimization and separation [15], we will address the *CG separation problem* in which we are given any point $x^* \in I\!R^n$ and we ask for a hyperplane separating x^* from P_1 (if any). Without loss of generality we can assume that $x^* \in P$, since all other points can be cut by simply enumerating the members of the original inequality system $Ax \leq b$, $x \geq 0$. Therefore, the separation problem we are actually interested in, called *CG-SEP* in the sequel, reads:

Definition 1. (CG-SEP) *Given any point $x^* \in P$ find (if any) a CG cut that is violated by x^*, i.e., find $u \in I\!R^m_+$ such that $\lfloor u^T A \rfloor x^* > \lfloor u^T b \rfloor$, or prove that no such u exists.*

It was proved by Eisembrand [10] that CG-SEP is NP-hard, so optimizing over P_1 also is.

In this paper we address the issue of evaluating the practical strength of P_1 in approximating P_I. Our approach is to model the rank-1 CG separation problem as a MIP model, which is then solved through a general-purpose MIP solver. As far as we know, this approach was never implemented (and evaluated computationally) by previous authors[1]. In Section 2 we describe our MIP model for CG separation. In Section 3 we investigate computationally the practical (and theoretical) benefits derived from using our separation tool. In particular, we report the optimal value over the first Chvàtal closure for a set of problems from MIPLIB 3.0 [4]. We also report, for the first time, the optimal solution of a very hard instance from MIPIB 2003 [1], namely *nsrand-ipx*, obtained by using our CG separation procedure to preprocess the original ILP model. Finally, we describe a new class of ATSP facets found with the help of our CG separation procedure.

2 How?

Given the input point $x^* \geq 0$ to be separated, CG-SEP calls for a CG cut $\alpha^T x \leq \alpha_0$ which is (maximally) violated by x^*, where $\alpha = \lfloor u^T A \rfloor$ and $\alpha_0 = \lfloor u^T b \rfloor$ for a certain $u \in I\!R_+$.

We observe that any variable x_j with $x_j^* = 0$ gives no contribution to the cut violation, hence there is no need to consider it explicitly in the separation model. Indeed, whenever $x_j^* = 0$ one can simply disregard x_j in the separation problem, and recompute the corresponding coefficient *a posteriori*, when the optimal u has been determined, as $\alpha_j := u^T A_j$ (no time-consuming lifting operations being needed). The same holds for variables at their upper bound in x^*, which can be complemented before separation (an operation that affects the right-hand-side vector b).

In addition, it is known that one can assume $u_i < 1$ in case the i-th row of (A, b) is integer, since in this case the CG cut associated with any $u \geq 0$ is easily seen to be the sum of the CG cut with u_i replaced by its fractional part, plus $\lfloor u_i \rfloor$ times the i-th constraint $a_i^T x \leq b_i$.

A major practical issue with CG separation is the strength of the returned cuts. As a matter of fact, several equivalent solutions of the separation problem (in its optimization version) typically exist, some of which produce very weak cuts for the ILP model (1). This is because the separation problem actually considers the face $F(x^*)$ of P_I where all the ILP constraints that are tight at x^* (including the variable bounds) are imposed as equalities. Hence for this face there exist several formulations of a same cut, which are equivalent for $F(x^*)$

[1] The fact that CG separation can be modeled as a MIP can instead be attributed to *folklore* and has been recently pointed out, e.g., by Caprara and Letchford [5] for the related class of *split cuts*.

but not for P_I. One therefore has to consider the issue of producing a violated cut that is as strong as possible with respect to P_I. In practice, this leads to the issue of producing "minimal" CG multiplier vectors with as few nonzero entries as possible.

In view of the considerations above, we propose the following MIP model for CG-SEP:

$$\max \ (\sum_{j \in J(x^*)} \alpha_j x_j^* - \alpha_0) - \sum_{i=1}^{m} w_i u_i \tag{6}$$

$$f_j = u^T A_j - \alpha_j, \quad \text{for } j \in J(x^*) \tag{7}$$

$$f_0 = u^T b - \alpha_0 \tag{8}$$

$$0 \leq f_j \leq 1 - \delta, \quad \text{for } j \in J(x^*) \cup \{0\} \tag{9}$$

$$0 \leq u_i \leq 1 - \delta, \quad \text{for } i = 1, \cdots, m \tag{10}$$

$$\alpha_j \text{ integer}, \quad \text{for } j \in J(x^*) \cup \{0\} \tag{11}$$

where $J(x^*) := \{j \in \{1, \cdots, n\} : x_j^* > 0\}$ denotes the support of x^* (possibly after having complemented same variables and updated b accordingly). In this model we introduced explicit slack variables $f_j = u^T A_j - \lfloor u^T A_j \rfloor$ to capture the coefficient fractionality, and required them to be in range $[0, 1 - \delta]$ for a small $\delta > 0$. (Some of these constraints are in fact redundant because of the objective function, but we observed that their incorporation into the model improves its numerical behavior.) In our implementation we chose the $\delta = 0.01$; this choice improves the numerical stability of our method, though it could affect the exact nature of our CG separation procedure in some pathological cases.

We also introduced the penalty term $-\sum_i w_i u_i$ in the objective function (6), where $w_i = 10^{-4}$ for all i, which is aimed at favoring the "minimality" of the CG multiplier vector u. It is worth observing that, because of equations (7)-(8), one can rewrite the violation term $\sum_{j \in J(x^*)} \alpha_j x_j^* - \alpha_0$ in (6) as $f_0 - \sum_{j \in J(x^*)} f_j x_j^* - \sum_{i=1}^{m} u_i s_i^*$, where $s^* = b - Ax^* \geq 0$. As a consequence, the penalty terms w_i are only relevant for tight constraints, i.e., when $s_i^* = 0$.

Model (6)-(11) can easily be adapted to deal with the case where the original model (1) contains free variables and/or equations. In addition, one can also allow for the presence of continuous variables[2]. Indeed, if a variable x_j is not restricted to be integer, one can still derive a valid cut by setting $\alpha_j = 0$ and imposing $u^T A_j \geq 0$ (or $u^T A_j = 0$ in case x_j is a free variable).

2.1 Implementation in a Pure Cutting-Plane Framework

We have embedded our CG cut separator into a pure cutting-plane framework where we keep generating violated CG cuts of rank 1 (with respect to the original formulation of the ILP model at hand), until we stop either because no such

[2] The mixed-integer case is however not covered in the original Chvàtal's work, and would be more appropriately addressed in the context of Gomory mixed-integer cuts.

violated cut exists (in which case we have optimized over the first closure), or because a time-limit condition is met. In our prototype implementation, we avoided a number of possible improvements consisting, e.g., in designing ad-hoc heuristics for our MIP separation, or in driving the MIP solver to produce a sequence of CG cuts with disjoint support. We only used the following simple mechanisms:

- When the LP relaxation of the original ILP model (1) is solved, we take all the violated Gomory fractional cuts that can be read from the current tableau, and skip CG separation; in principle this mechanism could also be applied in the subsequent iterations, provided that we skip the Gomory cuts having a non-zero CG multiplier for a constraint that was not in the original formulation, so as to ensure that all generated cuts are of rank 1.
- The MIP solver for CG separation is invoked with an initial lower bound of 0.01, meaning that we are only interested in CG cuts violated by more than 0.01.
- Each time the MIP solver for CG separation updates its incumbent solution, we store the corresponding CG cut in a pool, with the aim of adding it to the current ILP formulation right after the end of the separation phase. Because of the penalty term that appears in the MIP objective function (6), however, we found that it is useful to skip all the CG cuts with the same violation $\sum_{j \in J(x^*)} \alpha_j x_j^* - \alpha_0$, except the one with the sparsest support.
- The MIP execution for CG separation is stopped if either the optimal solution has been found, or a prefixed number τ of branching nodes has been explored after the last update of the incumbent solution, where $\tau = 1000$ if the violation of the incumbent is less than 0.2, and $\tau = 100$ if the violation is greater or equal to 0.2.

3 Why?

We next try to give concrete (i.e., supported by computational results) answers to some important questions related to the practical behavior of our MIP separator and to the theoretical effectiveness of rank-1 CG cuts, when applied to a wide range of ILP instances—both with and without a specific combinatorial structure.

Our implementation of the cutting-plane method uses the commercial software `ILOG-Cplex` 9.0.2 as the LP solver, whereas the separation problem is solved through `ILOG-Cplex` 9.0.2 MIP solver with "mip emphasis 4" parameter; see [13]. All computing times refer to a Pentium M 1.7 Ghz notebook with 512 MByte RAM.

3.1 Can We Solve Matching Problems?

In order to develop and test our code in a "familiar" setting, we addressed first the Edmonds-Johnson [9] *matching problem*, defined as:

$$\min\{c^T x : \ Ax = b, \ 0 \le x \le q, \ x \text{ integer}\} \tag{12}$$

where A, b, q are integer arrays of appropriate dimensions, and $\sum_{i=1}^{m} |a_{ij}| \leq 2$ holds for all $j = 1, \cdots, n$. A celebrated result of Edmonds [8, 9] says that P_1 is an integer polyhedron, i.e., P_1 and P_I coincide for matching problems. Moreover, undominated Chvàtal cuts (called *blossom* inequalities) only arise for $u \in \{0, 1/2\}^m$, and correspond to setting $u_i = 1/2$ for (a) the equations $a_i^T x = b_i$ associated with a certain set H (say) and (b) the upper-bound constraints $x_j \leq q_j$ associated with a possibly-empty set $T \subseteq \{1, \cdots, n\}$, such that $\sum_{i \in H} b_i + \sum_{j \in T} q_j$ is an odd number. Blossom inequalities can be separated in polynomial time by an ad-hoc procedure looking for a minimum-weight odd cut is a certain graph; see Padberg and Rao [16] and Letchford, Reinelt and Theis [14].

A relevant case of matching problems arises when A is the node-edge incidence matrix of a (multi-)graph $G = (V, E)$. Given edge costs c_e $(e \in E)$ and node values $b_v \geq 0$ $(v \in V)$, we call for a minimum-cost partial graph $G' = (V, M)$ of G where each node v has degree b_v. The *b-matching problem* can then be formulated as

$$\min\{c^T x : \sum_{e \in \delta(v)} x_e = b_v \text{ for } v \in V,\ 0 \leq x \leq 1,\ x \text{ integer}\} \tag{13}$$

Also belonging to the family of matching problems is the *T-join problem*, where each node v has an associated parity $b_v \in \{0, 1\}$ and one looks for a minimum-cost partial graph $G' = (V, M)$ of G where each node v has an odd degree if and only if $b_v = 1$. This problem can be formulated as the b-matching problem above, with the degree equations replaced by $\sum_{e \in \delta(v)} x_e - 2z_v = b_v$, z_v integer, for all $v \in V$.

Table 1 reports computational results on a set of 2-matching problems taken from TSPLIB: in all cases, our method was able to compute the optimal solution over the first Chvàtal closure, and produced an integer solution—as expected.

Table 1. 2-Matching Problems

ID	initial LB	Optimum	# iter.s	# cuts	CPU time
eil101	619.0	623.0	26	43	9.01
gr120	6,662.5	6,694.0	33	45	10.47
pr124	50,164.0	51,477.0	124	320	555.54
gr137	66,643.5	67,009.0	11	31	1.68
pr144	32,776.0	33,652.0	39	78	9.57
ch150	6,281.0	6,337.0	59	141	71.19
rat195	2,272.5	2,297.0	85	237	202.87
kroA200	27,053.0	27,426.0	26	84	10.93
kroB200	27,347.0	27,768.0	189	558	2,249.55
ts225	115,605.0	121,261.0	323	857	4,906.48
pr226	55,247.5	57,177.0	401	901	4,077.66
gr229	127,411.0	128,353.0	78	224	219.00
gil262	2,222.5	2,248.0	105	266	372.10
a280	2,534.0	2,550.0	52	104	40.21
lin318	38,963.5	39,266.0	292	768	6,103.32

Table 2. 2-Matching Problems: cut-and-branch approach

	ILOG-Cplex			cut-and-branch				
ID	% gap closed	nodes	time	# cuts	% gap closed	separation time	nodes	total time
pr124	100.0	43,125	104.17	116	62.1	27.96	1,925	37.51
kroB200	100.0	330,913	2,748.24	129	64.1	49.34	4,113	76.30
ts225	47.1	230,115	1h	250	80.7	164.77	13,552	352.35
pr226	55.0	288,901	1h	179	62.9	61.13	19,977	281.89
gr229	100.0	15,005	180.79	126	82.8	9.65	155	60.94
gil262	100.0	117,506	2,094.77	110	84.0	12.24	217	36.78
lin318	53.3	117,100	1h	187	64.9	110.69	25,953	933.97

It should be observed that some of these instances can be solved in a much shorter computing time by just applying ILOG-Cplex 9.0.2 MIP solver to the rank-0 model (13). However, for some hard instances our method also has a practical interest if used in a cut-and-branch approach where we abort our cut generation after the addition a certain number of CG cuts, and then switch to a commercial MIP solver for concluding the optimization. This is illustrated in Table 2, where some large 2-matching instances are solved through ILOG-Cplex MIP solver with and without the addition of an initial pool of CG cuts generated by 100 calls to our separation procedure.

3.2 How Tight Is the First Closure for MIPLIB Instances?

Table 3 reports the outcome of our experiments on a test-bed made by all pure-integer problems from MIPLIB 3.0 [4]. The cases where the MIP solver used within our separation procedure exceeded a time limit of 3 hours without providing a violated cut, are given in the bottom part of the table. In this situation, the percentage of integrality gap closed (*% gap closed*), computed as $100 - 100(opt\_value(P_I) - opt\_value(P_1))/(opt\_value(P_I) - opt\_value(P))$, is underestimated as we only have a lower bound on the optimal value $opt\_value(P_1)$ over the first closure.

In order to evaluate the viability of using our cut generator in a cut-and-branch approach akin to the one we described for 2-matching problems, we made the following experiment on the very hard ILP instance *harp2*. After having generated 211 rank-1 CG cuts (53 of which are tight at the end of the cutting-plane phase) in 100 rounds of separation, we switched to ILOG-Cplex MIP solver and obtained the optimal solution within 1,500 CPU seconds and 400K nodes (including both cut generation and branching), while ILOG-Cplex alone required more than 15,000 CPU seconds and 7M nodes to solve the original instance.

3.3 Beyond the First Closure?

The previous experiments show that optimizing over the first closure often gives a very tight approximation of the integer optimal value, though it may require

Table 3. General IPs of the MIPLIB 3.0

ID	Optimum	# iter.s	# cuts	% gap closed	time
air03	340,160.00	1	35	100.0	1.47
gt2	21,166.00	160	424	100.0	506.25
lseu	1,120.00	73	190	91.3	565.22
mitre	115,155.00	1,509	5,398	100.0	9,394.17
mod008	307.00	26	109	100.0	8.00
mod010	6,548.00	17	62	100.0	13.05
nw04	16,862.00	78	236	100.0	227.13
p0033	3,089.00	40	152	85.4	12.95
p0548	8,691.00	886	3,356	100.0	1,575.83
stein27	18.00	98	295	0.0	490.02
air04	56,137.00	128	901	$\geq$ 27.6	3h
air05	26,374.00	192	1,108	$\geq$ 15.5	3h
cap6000	-2,451,377.00	201	370	$\geq$ 26.9	3h
enigma	0.00	13,234	53,635	—	3h
fast0507	174.00	23	299	$\geq$ 4.7	3h
fiber	405,935.18	468	2,447	$\geq$ 98.5	3h
harp2	-73,899,798.00	1,081	1,661	$\geq$ 29.0	3h
l152lav	4,722.00	708	2,512	$\geq$ 69.2	3h
misc03	3,360.00	377	1,239	$\geq$ 51.2	3h
misc07	2,810.00	664	2,012	$\geq$ 16.1	3h
p0201	7,615.00	1,023	2,790	$\geq$ 60.5	3h
p0282	258,411.00	446	1,717	$\geq$ 99.9	3h
p2756	3,124.00	642	3,255	$\geq$ 69.2	3h
seymour	423.00	125	3,447	$\geq$ 23.5	3h
stein45	30.00	14,463	31,643	$\geq$ 0.0	3h

a large computing time. Besides the obvious issue of designing more specific separation procedures for CG cuts, we addressed the possibility of using our cutting plane method as a pre-processing tool, to be used to strengthen the user's formulation by possibly exploiting cuts of Chvàtal rank larger than 1. This idea was evaluated by comparing two different cut preprocessors, namely:

- *cpx*:Apply `ILOG-Cplex` 9.0.2 (with mip emphasis "move best bound") on the current ILP model, save the final root-node model (including the generated cuts) in a file, and repeat on the new model until a total time limit is exceeded.
- *cpx-cg*: Apply `ILOG-Cplex` 9.0.2 (with mip emphasis "move best bound") on the current ILP model, followed by 600 seconds of our CG separation procedure; then save in file the ILP model with all the cuts that are active in the last LP solution[3], and repeat on the new model until a total time limit is exceeded.

[3] As we no longer insist in staying within the first Chvàtal closure, CG cuts could be improved by deriving the stronger Gomory mixed-integer cut from the surrogate equations $^{T}A + \quad = \quad ^{T}b, \quad \geq 0$.

Figure 1 compares the behavior of the two methods above on a very hard instance from MIPLIB 2003, namely *timtab1* (viewed here as a pure ILP problem, as it is correct in this case). The horizontal axis counts the number of calls to the separation procedures (not the computing time nor the number of generated cuts), hence comparing the graph slopes can be misleading since every separation call produced about 10-50 cuts for *cpx*, but just 1-5 cuts for *cpx-cg*. Method *cpx* exhibits an early tailing-off phenomenon, while the growth for *cpx-cg* is more regular and produces a much better final lower bound.

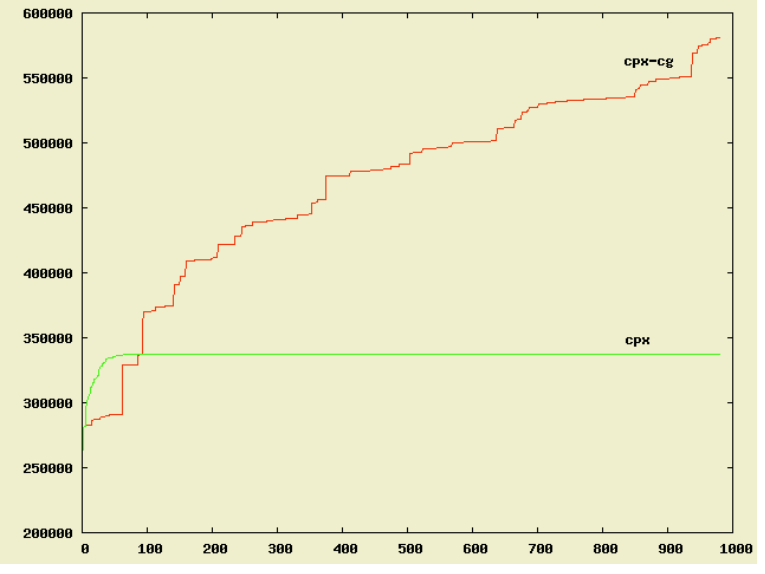


Fig. 1. Lower bounds provided by *cpx* and *cpx-cg* after each call of the separation procedures, for the hard MIPLIB instance *timtab1*

Remarkably, our cut preprocessor allowed us to find, for the first time [1], a provable optimal solution of value 51,200.00 for the very hard instance *nsrand-ipx* (in order to get a pure ILP, the objective function $x1$ was replaced by its expression given by the first constraint in the model.) To this end, we applied `ILOG-Cplex` 9.0.2 MIP solver to the tightened formulation returned by *cpx-cg* after 4800 seconds–this resulted in 23 new tight cuts that brought the initial LP bound from 49,667.89 to 50,665.71.

3.4 Can We Discover New Classes of Strong Inequalities?

The capability of separating over the CG cut class also has an interesting application in the off-line study of the facial structure of a specific problem. This is in the spirit of the widely-used approach of producing (through appropriate

software such as PORTA [6]) an explicit polyhedral description of very small instances of the problem at hand, but has the advantage of being applicable to instances of much larger size.

To illustrate a possible application to the *Asymmetric Travelling Salesman Problem* (ATSP), we made the following experiment. We set up a partial ATSP formulation including out- and in-degree equations, plus the subtour elimination constraints on 2-node sets, i.e., we addressed the NP-hard *Asymmetric Assignment Problem* (AAP) relaxation studied by Balas [2]. We then applied our separation procedure to a specific ATSP instance from TSPLIB, namely the 48-node instance *ry48p*, and stored the CG cuts we found along with the associated CG multipliers[4]. One of the returned CG cut had a violation of 0.75, and was associated with the following non-zero CG multipliers (reported in parenthesis): out-degree equation for nodes 21 (-0.5), 35 (0.5), 47 (-0.25), 2 (0.5), 26 (0.5), 12 (-0.25), 20 (-0.25), and 48 (-0.5); in-degree equations for nodes 5 (-0.5), 29 (-0.5), 35 (-0.5), 47 (0.25), 4 (-0.5), 42 (-0.5), 32 (0.5), 33 (0.25), and 20 (0.25); subtour elimination constraint on the 2-node sets $\{5, 29\}$ (0.5), $\{21, 47\}$ (0.25), and $\{35, 45\}$ (0.5). To simplify our analysis, we normalized the corresponding cut by adding 1 to all the negative CG multipliers, thus producing an equivalent CG cut in the so-called *h-canonical form* [3], i.e., with non-negative and relative prime coefficients, whose support leaves at least one node h isolated. This form immediately showed the presence of *clones* [3], that for the purpose of our analysis could be removed from the digraph. After renaming the nodes, we were left with a 8-node digraph and with the following CG multipliers: out-degree equation for nodes 1 (0.75), 3 (0.75), 4 (0.5), 5 (0.75), and 6 (0.5); in-degree equations for nodes 1 (0.25), 2 (0.5), 5 (0.25), 6 (0.5), and 7 (0.25); subtour elimination constraint on the 2-node sets $\{4, 5\}$ (0.25) and $\{6, 7\}$ (0.5). The associated CG cut is $\alpha^T x \leq \alpha_0$, where

$$\alpha = \begin{pmatrix} 0\,1\,0\,0\,1\,1\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,1\,0\,0\,1\,1\,1\,0 \\ 0\,1\,0\,0\,1\,1\,0\,0 \\ 1\,1\,0\,1\,0\,1\,1\,0 \\ 0\,1\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0 \end{pmatrix}; \alpha_0 = 5 \tag{14}$$

and it is easily shown (by computational methods) to be facet-defining for ATSP (and hence for AAP). Using *clique lifting* [3] we can then obtain a large class of ATSP facets, that to the best of our knowledge is new.

[4] For CG cuts in the form returned by our procedure, these multipliers could easily be recomputed *a posteriori*, by solving an LP.

References

1. T. Achterberg, T. Koch, A. Martin, The mixed integer programming library: MIPLIB 2003, `http://miplib.zib.de`.
2. E. Balas, The Asymmetric Assignment Problem and some new facets of the Traveling Salesman Polytope on a directed graph, *SIAM Journal on Discrete Mathematics* 2, 425–451, 1989.
3. E. Balas, M. Fischetti, A lifting procedure for the Asymmetric Traveling Salesman Polytope and a large new class of facet, *Mathematical Programming* 58, 325–352, 1993.
4. R.E. Bixby, S. Ceria, C.M. McZeal, M.W.P. Savelsbergh, MIPLIB 3.0, `http://www.caam.rice.edu/~bixby/miplib/miplib.html`.
5. A. Caprara, A.N. Letchford, On the separation of split cuts and related inequalities, *Mathematical Programming* 94, 279–294, 2003.
6. T. Christof, A. Löbel, PORTA - POlyhedron Representation Transformation Algorithm, `http://www.zib.de/Optimization/Software/Porta/`.
7. V. Chvátal, Edmonds polytopes and a hierarchy of combinatorial problems, *Discrete Mathematics* 4, 305–337, 1973.
8. J. Edmonds, Maximum matching and a polyhedron with {0,1}-vertices, *J. Res. Nat. Bur. Standards B* 69, 125–130, 1965.
9. J. Edmonds, H.L. Johnson, Matching: a well-solved class of integer linear programs, in R.K. Guy et al. (ed.s), *Combinatorial Structures and Their Applications*, Gordon and Breach, New York, 89–92, 1970.
10. F. Eisenbrand, On the membership problem for the elementary closure of a polyhedron, *Combinatorica* 19, 297-300, 1999.
11. R.E. Gomory, Outline of an algorithm for integer solutions to linear programs, *Bulletin of the AMS* 64, 275-278, 1958.
12. R.E. Gomory, An algorithm for integer solutions to linear programs, in R.L. Graves and P. Wolfe (ed.s) *Recent Advances in Mathematical Programming*, McGraw-Hill, New York, 1963.
13. ILOG Cplex 9.0.2: User's Manual and Reference Manual, ILOG, S.A., `http://www.ilog.com/`, 2004.
14. A.N. Letchford, G. Reinelt, D.O. Theis, A faster exact separation algorithm for blossom inequalities, in D. Bienstock and G. Nemhauser (ed.s), *Integer Programming and Combinatorial Optimization - IPCO 2004*, LNCS 3064, Springer-Verlag, Berlin Heidelberg, 196–205, 2004.
15. G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York, 1988.
16. M.W. Padberg, M.R. Rao, Odd Minimum Cut-Sets and b-Matchings, *Mathematics of Operations Research* 7, 67–80, 1982.

Sequential Pairing of Mixed Integer Inequalities

Yongpei Guan, Shabbir Ahmed, and George L. Nemhauser

School of Industrial & Systems Engineering,
Georgia Institute of Technology, Atlanta, GA, USA
{guanyp, sahmed, gnemhaus}@isye.gatech.edu

Abstract. We present a scheme for generating new valid inequalities for mixed integer programs by taking pair-wise combinations of existing valid inequalities. Our scheme is related to mixed integer rounding and mixing. The scheme is in general sequence-dependent and therefore leads to an exponential number of inequalities. For some important cases, we identify combination sequences that lead to a manageable set of non-dominated inequalities. We illustrate the framework for some deterministic and stochastic integer programs and we present computational results which show the efficiency of adding the new generated inequalities as cuts.

1 Introduction

We develop a scheme for generating new valid inequalities for mixed integer programs by taking pair-wise combinations of existing valid inequalities. Our scheme is related to the mixed integer rounding (MIR) procedure of Nemhauser and Wolsey [8, 9] and the mixing procedure of Günluk and Pochet [6]. We derive new inequalities iteratively by a very simple combination of two inequalities at a time, which we call *pairing*. As will be seen, the order in which the inequalities are paired is important since the resulting new inequalities depend on the order.

We describe the pairing procedure for pure integer programs and present a simple extension to MIPs in the next section. We study two structures in Sections 3 and 4 for which our pairing procedure gives nice results. We say that a set of inequalities is *nested* if component by component the coefficients in each successive inequality are no smaller than the coefficients in the previous inequalities. In the nested case, we show that there is a unique order for combining the inequalities that gives all of the nondominated inequalities that can be generated by the procedure. In this case, we obtain only a small number of inequalities and separation is fast. Moreover, we provide sufficient conditions for which the resulting inequalities are facet-defining. We say that a set of inequalities is *disjoint* if each integer variable appears in only one of the inequalities. Such disjoint sets arise in two-stage stochastic integer programming. Here we are again able to characterize the nondominated inequalities generated by the procedure, and we give a polynomial time separation algorithm. We also provide sufficient facet defining conditions.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 23–34, 2005.

Section 5 focuses on some applications of our procedure. In Section 6, we present computational results for nested and disjoint sets to demonstrate the strength of the inequalities in improving linear programming relaxation bounds. Proofs of all results in this paper are provided in [4].

2 The Pairing Scheme

Given a set of non-negative integer vectors $X \subset \mathbf{Z}^n_+$, a vector $a \in \mathbf{R}^{n+1}$ *defines* a valid inequality for X if

$$\sum_{j=1}^{n} a_j x_j - a_{n+1} \geq 0 \quad \text{for all } x \in X.$$

Given two such valid inequalities defined by vectors a and b, the one defined by a *dominates* the one defined by b if $a_j \leq b_j$ for all $j = 1, \ldots, n$ and $a_{n+1} \geq b_{n+1}$. We write $a \succeq b$.

The inequality $a \leq b$ for two vectors a and b of the same dimension is meant to hold component-wise. Similarly, $\min(a, b)$ and $\max(a, b)$ is understood to be carried out component-wise. For brevity, given a vector a and a scalar γ, we define $a + \gamma = a + \gamma\mathbf{1}$ and $\min\{a, \gamma\} = \min\{a, \gamma\mathbf{1}\}$, where $\mathbf{1}$ is a vector of ones of the same dimension as a.

Definition 1. *Given $a, b \in \mathbf{R}^{n+1}$ with $b_{n+1} \geq a_{n+1}$, we define the pairing of a and b as*

$$a \circ b = \min\{a + b_{n+1} - a_{n+1}, \max(a, b)\},$$

i.e., $(a \circ b)_{n+1} = b_{n+1}$ and

$$(a \circ b)_j = \begin{cases} a_j & \text{if } a_j \geq b_j \\ b_j & \text{if } a_j \leq b_j,\ b_j \leq a_j + b_{n+1} - a_{n+1} \\ a_j + b_{n+1} - a_{n+1} & \text{if } a_j \leq b_j,\ b_j \geq a_j + b_{n+1} - a_{n+1}, \end{cases}$$

for all $j = 1, \ldots, n$.

Theorem 1. *If $a, b \in \mathbf{R}^{n+1}$ define two valid inequalities for X, then $a \circ b$ defines a valid inequality for X.*

We note that there is a very simple and short direct proof of Theorem 1 [4], an alternate proof that uses the MIR procedure, and a third proof for the case of nonnegative coefficients that follows from Günluk and Pochet mixing.

Example. Consider the set

$$X = \{x \in \mathbf{Z}^3_+ : 3x_1 + 5x_2 \geq 3,\ 5x_2 + 4x_3 \geq 5\}.$$

The two original inequalities for X are defined by $a = (3, 5, 0, 3)$ and $b = (0, 5, 4, 5)$. The valid inequality defined by $a \circ b$ is

$$3x_1 + 5x_2 + 2x_3 \geq 5. \tag{1}$$

To see that (1) can be useful, note that it cuts off the fractional point $(0, 3/5, 1/2)$ which is feasible to the LP relaxation of X.

The pairing scheme can be easily applied to mixed-integer sets. The pair $(a, g) \in \mathbf{R}^{n+1} \times \mathbf{R}^p$, defines a valid inequality for a mixed-integer set $Y \subset \mathbf{Z}^n_+ \times \mathbf{R}^p_+$ if

$$\sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{p} g_j y_j \geq a_{n+1} \quad \text{for all } (x, y) \in Y.$$

Corollary 1. *If (a^1, g^1) and (a^2, g^2) define two valid inequalities for Y, then $(a^1 \circ a^2, \max\{g^1, g^2\})$ defines a valid inequality for Y.*

Note that the standard disjunctive inequality (see, e.g. [8]), obtained from the inequalities (a^1, g^1) and (a^2, g^2) for Y,

$$\sum_{i=1}^{n} \max\{a^1_i, a^2_i\} x_i + \sum_{j=1}^{p} \max\{g^1_j, g^2_j\} y_j \geq \min\{a^1_{n+1}, a^2_{n+1}\},$$

is dominated by the pairing inequality in Corollary 1.

We now consider the pairing inequalities obtained from a set of inequalities. Suppose we have K valid inequalities for X defined by the vectors $\{a^1, \ldots, a^K\} \subset \mathbf{R}^{n+1}$. Given a subset of these K vectors, we can obtain new valid inequalities by carrying out a sequence of pairing operations. For example, the valid inequality defined by the vector $((a^{k_1} \circ a^{k_2}) \circ (a^{k_2} \circ a^{k_3})) \circ a^{k_4}$ is obtained from $\{a^{k_1}, a^{k_2}, a^{k_3}, a^{k_4}\}$ with the parentheses distinguishing the sequence in which the pairings are carried out. Since the $\circ$ operation is not associative, the valid inequalities obtained from a given set of vectors *depends on the sequence* in which the pairings are done. Thus from the set of K valid inequalities defined by $\{a^1, \ldots, a^K\}$ we can generate an exponential number of inequalities depending on the subset of valid inequalities chosen and the sequence in which they are mixed. A key problem is to identify pairing sequences that lead to good sets of valid inequalities, i.e., strong inequalities over which separation can be done efficiently.

In the following two sections, we investigate a pairing sequence that leads to two such families of inequalities. This pairing sequence is defined by

Definition 2. *Given a finite set of vectors, i.e., $A = \{a^1, \ldots, a^K\}$, where $a^1_{n+1} \leq a^2_{n+1} \leq, \ldots, \leq a^K_{n+1}$, we define sequential pairing of the vectors in A by*

$$\Delta(A) = ((\ldots((a^1 \circ a^2) \circ a^3) \circ \ldots) \circ a^K).$$

3 The Nested Case

Consider a set $A = \{a^1, \ldots, a^K\} \subset \mathbf{R}^{n+1}$ such that $a^1 \leq \ldots \leq a^K$. We say that the valid inequalities defined by the vectors in A are (or the set A itself

is) *nested*. Here we consider mixed integer systems where the coefficients of the integer variables are nested. Nested sets arise, for example, in the dynamic knapsack problem considered by Loparic, Marchand and Wolsey [7] where the feasible region is given by

$$X = \Big\{(x, y) \in \{0,1\}^n \times \mathbf{R}_+ : \quad \sum_{j=1}^{i} a_j x_j + y \geq \sum_{j=1}^{i} d_j, \quad i = 1, \ldots, n\Big\}, \qquad (2)$$

with $a \in \mathbf{R}^n_+$ and $d \in \mathbf{R}^n_+$. Here, y is a continuous inventory variable, $x_j \in \{0,1\}$ represents whether the amount a_j is produced in period j, and d_j is the demand in period j.

Let $A_k = \{a^1, \ldots, a^k\}$ for $k = 1, \ldots, K$, and let $\Phi(A) \in \mathbf{R}^{n+1}$ be a vector obtained by an arbitrary sequence of pairings of the vectors in A. Next, we show that $\Delta(A) \succeq \Phi(A)$.

Theorem 2. *If $A = \{a^1, \ldots, a^K\}$ is nested, then*

$$\Delta(A) = \min\{a^1 + a^K_{n+1} - a^1_{n+1}, a^2 + a^K_{n+1} - a^2_{n+1}, \ldots, a^{K-1} + a^K_{n+1} - a^{K-1}_{n+1}, a^K\}.$$

Theorem 3. *If A is nested, then $\Delta(A) \succeq \Phi(A)$ for any $\Phi(A)$.*

Lemma 1. *If $A = \{a^1, \ldots, a^K\}$ is nested and $B \subset A$ is such that $a^K \in B$, then $\Delta(A) \succeq \Delta(B)$.*

Combining Theorem 3 and Lemma 1, we obtain

Theorem 4. *Let $A = \{a^1, \ldots, a^K\}$ be nested. All the non-dominated inequalities obtained by pairings of the vectors in A are contained in the set* $\cup_{k=1}^{K} \{\Delta(A_k)\}$.

Hence there are at most K non-dominated inequalities.

Now we give sufficient conditions for the inequalities in $\cup_{k=1}^{K} \Delta(A_k)$ to be facet-defining. Let $A = \{a^1, \ldots, a^K\} \in \mathbf{R}^{n+1}$ be a nested set such that $a^i \geq 0$ for all $i = 1, \ldots, K$, and consider the mixed 0-1 set (with one continuous variable):

$$X = \{(x, y) \in \{0,1\}^n \times \mathbf{R}_+ : \ \sum_{j=1}^{n} a^i_j x_j + y \geq a^i_{n+1}, \ i = 1, \ldots, K\}.$$

Without loss of generality, we assume that $a^i_j \leq a^i_{n+1}$ for all $j = 1, \ldots, n$ and $i = 1, \ldots, K$, since otherwise the coefficients can be strengthened to $a^i_j = a^i_{n+1}$. Let $A_i = \{a^1, \ldots, a^i\}$ for $i = 1, \ldots, K$, and $\Delta^i = \Delta(A_i)$.

Theorem 5. *Given $i \in \{1, \ldots, K\}$, the sequential pairing inequality*

$$\sum_{j=1}^{n} \Delta^i_j x_j + y \geq a^i_{n+1}$$

is facet-defining if, for all $k \in \{i, i+1, \ldots, K\}$,

(a) there exists $j^* \in \{1, \ldots, n\}$ such that $\Delta^i_{j^*} + a^k_{n+1} - a^i_{n+1} \leq a^k_{j^*}$, and
(b) $\sum_{j \in Z(i)} a^k_j \geq a^k_{n+1} - a^i_{n+1}$ where $Z(i) = \{j \in \{1, \ldots, n\} : a^i_j = 0\}$.

4 The Disjoint Case

A set $A = \{a^1, \ldots, a^K\} \subset \mathbf{R}^{n+1}$ satisfying

1. $a^k \geq 0$ for all $k = 1, \ldots, K$,
2. for any two vectors a^l and a^m, $a^l_j a^m_j = 0$ for $j = 1, \ldots, n$, and
3. $a^1_{n+1} \leq a^2_{n+1} \leq \ldots \leq a^K_{n+1}$.

is said to be *disjoint*. Here we consider mixed integer systems where the coefficients of the integer variables are disjoint. An example is the deterministic equivalent formulation of a two-stage stochastic program with integer second stage variables [3]

$$\begin{aligned} \min \ & c^T y + \sum_{s=1}^{S} p_s q_s^T x_s \\ & y \in Y \subseteq \mathbf{R}_+^{n_1 - p_1} \times \mathbf{Z}_+^{p_1} \\ & T_s y + W_s x_s \geq h_s \qquad s = 1, \ldots, S \\ & x_s \in \mathbf{Z}_+^{n_2} \qquad\qquad\quad s = 1, \ldots, S. \end{aligned}$$

In the above model, there are two sets of decision variables. The first-stage variables y are decided prior to a scenario s of realizations of the uncertain problem parameters (q_s, T_s, W_s, h_s). The second-stage decisions x_s constitute "recourse" actions corresponding to the scenario s realized. A scenario s occurs with probability p_s, and the objective is to minimize the sum of first-stage and expected second-stage costs. Note that the second-stage variables constitute a disjoint system.

Theorem 6. *If $A = \{a^1, \ldots, a^K\}$ is disjoint, then*

$$\Delta(A) = a^1 + \sum_{i=2}^{K} \min\{a^i_{n+1} - a^{i-1}_{n+1}, a^i\}.$$

As before, we let $\Phi(A) \in \mathbf{R}^{n+1}$ be a vector obtained by an arbitrary sequence of pairings of the vectors in A.

Theorem 7. *If $A = \{a^1, \ldots, a^K\}$ is disjoint, then for any $\Phi(A)$, there exists $\widehat{A} \subseteq A$ with $a^K \in \widehat{A}$ such that*

$$\Delta(\widehat{A}) \succeq \Phi(A).$$

As a consequence of Theorem 7, among all inequalities obtained by pairings of the vectors in a disjoint set A, it is sufficient to consider the inequalities corresponding to the $2^K - 1$ vectors in $C = \{\Delta(\widehat{A}) : \ \widehat{A} \subseteq A, \widehat{A} \neq \emptyset\}$.

Even though it suffices to consider the inequalities defined by the set C, the number of such inequalities is exponential in K. Here we present a polynomial time separation algorithm based on solving shortest path problems on a directed

graph G with nodes $\mathcal{N} = \{0, 1, \ldots, K\}$ and arcs (i, j) for all i and $j > i$. Given a point x^*, the separation problem of determining whether there exists any violated pairing inequalities can be reduced to finding a shortest path from node 0 to node k for $1 \leq k \leq K$ where the length of arc (i, j) is given by $\sum_{r=1}^{n} \min\{a_r^j, a_{n+1}^j - a_{n+1}^i\} x_r^*$ for $i > 0$ and $\sum_{r=1}^{n} a_r^j x_r^*$ for $i = 0$. Using Dijkstra's algorithm the separation problem can be solved in $O(K^2)$ time and we can find as many as K violated inequalities from the shortest paths from 0 to k for $k = 1, \ldots, K$.

Now we give sufficient conditions for the inequalities in C to be facet-defining. Let $A = \{a^1, \ldots, a^K\} \in \mathbf{R}^{n+1}$ be a disjoint set, and consider the mixed 0-1 set :

$$X = \{(x, y) \in \{0, 1\}^n \times \mathbf{R}_+ : \sum_{j=1}^{n} a_j^i x_j + y \geq a_{n+1}^i,\ i = 1, \ldots, K\},$$

with one continuous variable. Without loss of generality, as in the nested set case, we assume that $a_j^i \leq a_{n+1}^i$ for all $j = 1, \ldots, n$ and $i = 1, \ldots, K$. We also assume that $\sum_{j=1}^{n} a_j^i \geq a_{n+1}^i,\ \ i = 1, \ldots, K$, since otherwise, we can replace y by $y + (a_{n+1}^i - \sum_{j=1}^{n} a_j^i)$. Consider $\widehat{A} = \{a^{q_1}, \ldots, a^{q_Q}\} \subseteq A$. Denote $\mathcal{Q} = \{q_1, \ldots, q_Q\}$ and, for brevity, let $q = q_1$, $Q = q_Q$. Define $\widehat{\Delta} = \Delta(\widehat{A})$, where the jth element $\widehat{\Delta}_j$ is given by $\widehat{\Delta}_j = \min\{a_{n+1}^{r(j)} - a_{n+1}^{c(r(j))}, a_j^{r(j)}\}$, with $r(j) = \{i \in \{1, \ldots, n\} : a_j^i > 0\}$ for $j = 1, \ldots, n$ and $c(i) = \text{argmax}\{k \in \mathcal{Q} : k < i\}$ for all $i \in \mathcal{Q}$.

Theorem 8. *Given $\widehat{A} \subseteq A$ and the corresponding index set $\mathcal{Q}$, the sequential pairing inequality*

$$\sum_{j=1}^{n} \widehat{\Delta}_j x_j + y \geq a_{n+1}^Q$$

is facet-defining if

(a) $\max\{a_j^i : j \in \{1, \ldots, n\}\} \geq \max\{a_j^q : j \in \{1, \ldots, n\}\} + a_{n+1}^i - a_{n+1}^q$, *for all* $i \in \mathcal{Q}$.
(b) $\sum_{j=1}^{n} a_j^i \geq a_{n+1}^i - a_{n+1}^Q + a_k^i$, *for all* $k \in \{1, \ldots, n\}$ *and* $i \in \{Q+1, \ldots, K\}$.

5 Applications

Dynamic Knapsack Sets: Consider the Dynamic knapsack set X given by (2) with $a \in \mathbf{R}_+^n$ and $d \in \mathbf{R}_+^n$. Let $d_{ij} = \sum_{k=i}^{j} d_k$, Loparic et al. [7] proved that the inequality

$$y + \sum_{j=1}^{i} \min\{a_j, d_{ji}\} x_j \geq d_{1i} \tag{3}$$

is valid for $\text{conv}(X)$ for $i = 1, \ldots, n$, and facet-defining when $i = n$. Dynamic knapsack sets are nested. Applying the pairing sequence Δ to the first i inequalities of (2) gives the inequality (3).

Star Inequalities: Given the mixed 0-1 set

$$Y = \{(x, y) \in \{0,1\}^n \times \mathbf{R} : \ a_i x_i + y \leq u \ \ i = 1, \ldots, n\},$$

with $a_1 \leq a_2 \leq \ldots \leq a_n$, Atamtürk et al. [1] showed that the *star* inequality

$$\sum_{i=1}^{n} \bar{a}_i x_i + y \leq u,$$

where $\bar{a}_1 = a_1$ and $\bar{a}_i = a_i - a_{i-1}$ for $i = 2, \ldots, n$, is valid for Y. Note that the inequalities that define Y form a disjoint set. By complementing the binary variables and applying the pairing sequence Δ, we obtain the *star* inequality.

Deterministic Lot-Sizing: The deterministic uncapacitated lot-sizing problem is to minimize total production and inventory holding cost while satisfying demand over a finite discrete-time planning horizon. Let y_i be the production in period i, $x_i \in \{0,1\}$ indicate if there is a production set-up in period i, d_i be the demand in period $i \in \{1, \ldots, n\}$, and $d_{st} = \sum_{i=s}^{t} d_i$. The feasible solution set of the lot-sizing problem is

$$X_{\text{LS}} = \{(x, y) \in \{0,1\}^n \times \mathbf{R}^n_+ : \ \sum_{j=1}^{i} y_j \geq d_{1i}, \ 0 \leq y_i \leq d_{in} x_i, \ \ i = 1, \ldots, n\}.$$

Barany et al. [2] described the convex hull of X_{LS} by introducing the (ℓ, S) inequalities

$$\sum_{i \in S} y_i + \sum_{i \in L \setminus S} d_{i\ell} x_i \geq d_{1\ell} \tag{4}$$

for $1 \leq \ell \leq n$, $L = \{1, \ldots, \ell\}$ and $S \subseteq L$.

We now show that the pairing scheme can generate all of the (ℓ, S) inequalities. For given ℓ and S , we use the constraints $\sum_{j \leq k} y_j \geq d_{1k}$ for each $k \leq \ell$ and $y_j \leq d_{jn} x_j$ for each $j \in \{1, \ldots, n\}$ to obtain the inequalities

$$\sum_{j \in S_k} y_j + \sum_{j \in L_k \setminus S_k} d_{jn} x_j \geq d_{1k} \text{ for each } k \leq \ell, \tag{5}$$

where $L_k = \{1, 2, \ldots, k\}$ and $S_k = S \cap L_k$. The family of inequalities (5) is nested (note here $1 \in S_k$ for each $k \leq \ell$). By Theorem 2, sequential pairing provides the (ℓ, S) inequality in (4) since we have $\Delta^\ell_j = \min\{d_{1\ell} - d_{11}, \ldots, d_{1\ell} - d_{1(j-1)}, d_{jn} + d_{1\ell} - d_{1j}, \ldots, d_{jn}\} = d_{1\ell} - d_{1(j-1)} = d_{j\ell}$ corresponding to each $j \in L \setminus S$.

Stochastic Lot-Sizing: The stochastic uncapacitated lot-sizing problem is the stochastic programming extension of the deterministic formulation. Instead of deterministic cost and demand information for each time period, the problem parameters are random and evolve as discrete time stochastic processes with a finite probability space. A scenario tree is used to model this information where

each node i in stage t of the tree represents a possible state of the system. For each node i, let $\mathcal{T}(i) = (\mathcal{V}(i), \mathcal{E}(i))$ be the subtree containing all descendants of node i, $\mathcal{L}(i)$ be the leaf nodes of the subtree $\mathcal{T}(i)$, $\mathcal{P}(i,j)$ be the set of nodes on the path from node i to node j and $d_{ij} = \sum_{k \in \mathcal{P}(i,j)} d_k$, where d_i represents the demand in period $t(i)$ for node i. For brevity, let $\mathcal{T} = \mathcal{T}(0), \mathcal{V} = \mathcal{V}(0), \mathcal{L} = \mathcal{L}(0)$ and $\mathcal{P}(i) = \mathcal{P}(0,i)$.

Let y_i be the production and x_i be the indicator variable for a production set-up in period $t(i)$ corresponding to the state defined by node i. The feasible solution set of the stochastic lot-sizing problem [5] is

$$X_{\text{SLS}} = \{(x,y) \in \{0,1\}^n \times \mathbf{R}^n_+ : \sum_{j \in \mathcal{P}(i)} y_j \geq d_{0i},\ 0 \leq y_i \leq M_i x_i, \quad i \in \mathcal{V}\},$$

where $M_i = \max_{j \in \mathcal{L}(i)} d_{ij}$ is an upper bound on y_i.

Guan et al. [5] have developed a family of valid inequalities for X_{SLS} called the $(\mathcal{Q}, S_{\mathcal{Q}})$ inequalities. Consider a set of nodes $\mathcal{Q} = \{1, 2, \ldots, Q\} \subset \mathcal{V}$, such that $d_{01} \leq d_{02} \leq \ldots \leq d_{0Q}$ and $\{m, m+1, \ldots, n-1, n\} \subseteq \mathcal{Q}(i)$ if $m < n$ and $m, n \in \mathcal{Q}(i)$, where $\mathcal{Q}(i) = \mathcal{Q} \cap \mathcal{V}(i)$. Let $\mathcal{V}_{\mathcal{Q}} = \cup_{i \in \mathcal{Q}} \mathcal{P}(i)$ and for each $i \in \mathcal{V}_{\mathcal{Q}}$ let

$$\begin{aligned}
\overline{D}_{\mathcal{Q}}(i) &= \max\{d_{0j} :\ j \in \mathcal{Q}(i)\},\\
\widetilde{D}_{\mathcal{Q}}(i) &= \begin{cases} 0, & \text{if } \{j : j \in \mathcal{Q} \setminus \mathcal{Q}(i) \text{ such that } d_{0j} \leq \overline{D}_{\mathcal{Q}}(i)\} = \emptyset \\ \max\{d_{0j} :\ j \in \mathcal{Q} \setminus \mathcal{Q}(i) \text{ such that } d_{0j} \leq \overline{D}_{\mathcal{Q}}(i)\}, & \text{otherwise,} \end{cases}\\
M_{\mathcal{Q}}(i) &= \max\{d_{ij} : j \in \mathcal{Q}(i)\}, \text{ and}\\
\delta_{\mathcal{Q}}(i) &= \min\left\{\overline{D}_{\mathcal{Q}}(i) - \widetilde{D}_{\mathcal{Q}}(i), M_{\mathcal{Q}}(i)\right\}.
\end{aligned}$$

Then, given $S_{\mathcal{Q}} \subseteq \mathcal{V}_{\mathcal{Q}}$ and $\overline{S}_{\mathcal{Q}} = \mathcal{V}_{\mathcal{Q}} \setminus S_{\mathcal{Q}}$, the $(\mathcal{Q}, S_{\mathcal{Q}})$ inequality

$$\sum_{i \in S_{\mathcal{Q}}} y_i + \sum_{i \in \overline{S}_{\mathcal{Q}}} \delta_{\mathcal{Q}}(i) x_i \geq M_{\mathcal{Q}}(0) \tag{6}$$

is valid for X_{SLS}.

Although the inequalities that define X_{SLS} are neither nested or disjoint, we can generate all $(\mathcal{Q}, S_{\mathcal{Q}})$ inequalities by pairings as follows. For a given $\mathcal{Q}$ and $S_{\mathcal{Q}}$, we use pairings, as in the deterministic lot-sizing case, to generate the (ℓ, S) inequalities

$$\sum_{j \in \mathcal{P}(i) \cap S_{\mathcal{Q}}} y_j + \sum_{j \in \mathcal{P}(i) \cap \overline{S}_{\mathcal{Q}}} d_{ji} x_j \geq d_{0i}$$

corresponding to $\mathcal{P}(i)$ for each $i \in \mathcal{Q}$. Then, we use sequential pairing of the above (ℓ, S) inequalities for $i = 1, \ldots, Q$ and obtain the $(\mathcal{Q}, S_{\mathcal{Q}})$ inequality (6).

6 Computational Experiments

In this section we provide some numerical results to demonstrate the computational effectiveness of the pairing scheme on randomly generated instances of

mixed-integer programs with nested and disjoint sets of constraints. All computations have been carried out on a Linux workstation with dual 2.4 GHz Intel Xeon processors and 2 GB RAM using CPLEX 8.1.

For the nested case, we generated random instances of the model

$$\begin{aligned}
\min \sum_{j=1}^{mn} c_j x_j + \sum_{k=1}^{p} h_k y_k & \\
\sum_{j=1}^{in} a_j^i x_j + \sum_{k=1}^{p} g_k^i y_k \geq b_i & \; i = 1, \ldots, m \\
x_j \in \{0,1\} & \quad j = 1, \ldots, mn \\
y_k \geq 0 & \quad k = 1, \ldots, p.
\end{aligned}$$

This model has n additional binary variables in each successive row, with a total of mn binary variables and p continuous variables. The constraint coefficients and the right-hand sides were generated such that these form a nested system and were uniformly distributed within the interval $[50, 75]$ and $[50, 100]$, respectively. The objective function coefficients were uniformly distributed within the interval $[10, 100]$. In Table 1 we present computational results for $p \in \{1, 2, 3\}$, $n \in \{1, 2, 3\}$ and $m \in \{10, 20, 40\}$. For each combination of m, n, and p, we tested five instances and report the average objective function value in the column labelled "OptVal." The row labelled "LP" provides the average optimal objective value of the linear programming relaxation without any cuts; the row labelled "LP+CUTS"(LPC) provides the average optimal objective value after adding all inequalities obtained through pairing as cuts, which can be done since the total number of cuts is small and equal to the number of rows; and the row labelled "IP" provides the optimal value of the corresponding integer programming problem. The column labelled "Gap" provides the percentage LP relaxation gap, computed as (IP-LP)/LP $\times$ 100% and (IP-LPC)/LPC $\times$ 100%. We observe that the cuts yield significant improvements. In 13 of the 27 cases, the gap is reduced to 0% from over 10%. In all but three of the cases, the gap is reduced by more than half.

For the disjoint case, we generated random instances of the model

$$\begin{aligned}
\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_j^i x_j^i + \sum_{k=1}^{p} h_k y_k & \\
\sum_{j=1}^{n} a_j^i x_j^i + \sum_{k=1}^{p} g_k y_k \geq b_i & \; i = 1, \ldots, m \\
x_j^i \in \{0,1\} & \quad j = 1, \ldots, n, i = 1, \ldots, m \\
y_k \geq 0 & \quad k = 1, \ldots, p.
\end{aligned}$$

Each row of this model has n independent binary variables giving rise to a disjoint system involving a total of mn binary variables. A total of p continuous variables couple the binary variables together. The constraint coefficients and the right-hand sides were generated uniformly within the interval $[40, 120]$ and

Table 1. Computational Results for the Nested Case

			= 10		= 20		= 40	
			OptVal	Gap	OptVal	Gap	OptVal	Gap
1	1	LP	100.59	19.58%	51.10	13.97%	24.43	22.18%
		LP+CUTS	114.41	8.53%	59.40	0.00%	31.18	0.67%
		IP	125.08		59.40		31.39	
1	2	LP	65.88	23.63%	48.87	13.25%	21.21	15.99%
		LP+CUTS	77.80	9.82%	56.33	0.00%	25.02	0.89%
		IP	86.27		56.33		25.25	
1	3	LP	39.71	19.83%	48.19	14.50%	21.43	14.56%
		LP+CUTS	43.27	12.64%	56.36	0.00%	24.95	0.51%
		IP	49.53		56.36		25.08	
2	1	LP	23.00	4.61%	31.47	10.38%	65.86	13.05%
		LP+CUTS	24.11	0.00%	35.12	0.00%	75.75	0.00%
		IP	24.11		35.12		75.75	
2	2	LP	22.62	9.50%	31.45	11.65%	58.21	15.77%
		LP+CUTS	24.99	0.00%	35.60	0.00%	66.96	3.11%
		IP	24.99		35.60		69.11	
2	3	LP	22.02	7.92%	31.42	13.90%	56.89	15.39%
		LP+CUTS	23.92	0.00%	36.49	0.00%	63.95	4.87%
		IP	23.92		36.49		67.23	
3	1	LP	20.28	19.45%	21.99	30.57%	69.13	14.96%
		LP+CUTS	24.18	3.95%	28.03	11.52%	81.29	0.00%
		IP	25.18		31.68		81.29	
3	2	LP	17.05	28.54%	20.39	27.60%	64.66	13.35%
		LP+CUTS	20.47	14.20%	22.81	18.99%	74.62	0.00%
		IP	23.86		28.16		74.62	
3	3	LP	18.99	25.28%	20.06	29.40%	64.13	11.93%
		LP+CUTS	22.52	11.41%	22.74	19.96%	72.82	0.00%
		IP	25.42		28.41		72.82	

$[100, 125]$ respectively. The objective function coefficients were uniformly distributed within the interval $[10, 100]$ for the continuous variables and within the interval $[10/m, 100/m]$ for the binary variables. In Table 2, we present computational results corresponding to $p \in \{1, 2, 3\}$, $n \in \{1, 2, 3\}$ and $m \in \{10, 20, 40\}$. As before, we report averages over five random instances for each combination of m, n and p. In this case, we use the shortest path separation routine described in Section 4 to add only violated cuts. The average number of cuts added is reported in the row labelled "# CUTS." Once again, we observe that the cuts yield significant improvements. In 6 of the 27 cases, the gap is reduced to 0%. In 19 of the 27 cases, the gap is reduced by more than half. The number of cuts ranges from 30, on average for 10 rows, to 491, on average for 40 rows.

Table 2. Computational Results for the Disjoint Case

			= 10		= 20		= 40	
			OptVal	Gap	OptVal	Gap	OptVal	Gap
		LP	1082.24	11.62%	703.83	9.03%	874.35	12.57%
1	1	LP+CUTS	1099.88	10.17%	729.58	5.71%	928.66	7.14%
		IP	1224.47		773.73		1000.08	
		# CUTS	45		163		925	
		LP	585.63	38.48%	538.49	25.20%	702.46	29.22%
1	2	LP+CUTS	793.18	16.68%	655.58	8.93%	946.30	4.65%
		IP	952.00		719.89		992.45	
		# CUTS	32		162		1317	
1	3	LP	410.88	26.45%	446.83	23.08%	559.81	24.67%
1	3	LP+CUTS	480.88	13.92%	452.76	22.05%	619.81	16.59%
		IP	558.66		580.87		743.13	
		# CUTS	10		27		124	
		LP	685.99	8.36%	388.66	5.96%	497.25	7.69%
2	1	LP+CUTS	693.98	7.29%	400.19	3.17%	522.35	3.03%
		IP	748.58		413.29		538.67	
		# CUTS	34		74		341	
		LP	507.76	33.72%	356.65	14.23%	464.42	13.56%
2	2	LP+CUTS	710.69	7.22%	415.34	0.12%	529.88	1.38%
		IP	766.03		415.83		537.30	
		# CUTS	39		119		530	
		LP	400.18	20.50%	339.61	12.71%	437.98	13.60%
2	3	LP+CUTS	448.58	10.89%	357.13	8.21%	470.21	7.25%
		IP	503.38		389.06		506.95	
		# CUTS	20		63		347	
		LP	533.31	2.84%	285.17	2.09%	387.16	4.80%
3	1	LP+CUTS	542.35	1.19%	291.25	0.00%	406.67	0.00%
		IP	548.88		291.25		406.67	
		# CUTS	25		34		173	
		LP	433.80	21.02%	280	3.86%	375.75	6.84%
3	2	LP+CUTS	540.33	1.63%	291.23	0.00%	403.35	0.00%
		IP	549.27		291.23		403.35	
		# CUTS	45		59		289	
		LP	420.45	14.19%	279.87	4.38%	360.44	9.80%
3	3	LP+CUTS	459.61	6.19%	292.69	0.00%	399.61	0.00%
		IP	489.95		292.69		399.61	
		# CUTS	24		67		375	

7 Conclusions

We have developed a new and very simple way of pairwise combining linear inequalities for MIPs to obtain new linear inequalities. These new inequalities can be useful in tightening the LP relaxation for general MIPs. The order in which the inequalities are combined can have a significant impact on the results. For some structured systems, we provided combination orders that are optimal in the

sense that no other combination order cannot dominate the set of inequalities given by the optimal order. These structures arise in multi-period MIPs. We discussed applications of these structures to deterministic and stochastic lot-sizing problems. One of our goals is to apply the procedure to general multi-period stochastic MIPs. To do this we need to generalize the structures considered in this paper to scenario trees. We are currently developing these results.

Acknowledgements

This research has been supported by the National Science Foundation awards DMI-0100020, DMI-0121495, and DMI-0133943. The authors thank Alper Atamtürk and Oktay Günluk for helpful comments.

References

1. A. Atamtürk, G. L. Nemhauser and M. W. P. Savelsbergh. The mixed vertex packing problem. *Mathematical Programming*, 89:35–53, 2000.
2. I. Barany, T. Van Roy, and L. A. Wolsey. Strong formulations for multi-item capacitated lot sizing. *Management Science*, 30:1255–1262, 1984.
3. J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer, 1997.
4. Y. Guan, S. Ahmed, and G. L. Nemhauser. Sequential pairing of mixed integer inequalities. Technical Report No. TLI-04-04, School of Industrial & Systems Engineering, Georgia Institute of Technology, 2004.
5. Y. Guan, S. Ahmed, G. L. Nemhauser, and A. J. Miller. A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. To appear in *Mathematical Programming*, 2005.
6. O. Günluk and Y. Pochet. Mixing mixed-integer inequalities. *Mathematical Programming*, 90:429–457, 2001.
7. M. Loparic, H. Marchand and L. A. Wolsey. Dynamic knapsack sets and capacitated lot-sizing. *Mathematical Programming*, 95:53–69, 2003.
8. G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, 1988.
9. G. L. Nemhauser and L. A. Wolsey. A recursive procedure for generating all cuts for 0-1 mixed integer programs. *Mathematical Programming*, 46:379–390, 1990.

Approximate Min-max Relations for Odd Cycles in Planar Graphs

Samuel Fiorini[1,⋆], Nadia Hardy[2,⋆⋆], Bruce Reed[2,⋆⋆⋆], and Adrian Vetta[2,†]

[1] GERAD, Montreal, Quebec, Canada and Université Libre de Bruxelles, Brussels, Belgium
sfiorini@ulb.ac.be
[2] McGill University, Montreal, Quebec, Canada
{hardy, vetta}@math.mcgill.ca
breed@cs.mcgill.ca

Abstract. We study the ratio between the minimum size of an odd cycle vertex transversal and the maximum size of a collection of vertex-disjoint odd cycles in a planar graph. We show that this ratio is at most 10. For the corresponding edge version of this problem, Král and Voss [7] recently proved that this ratio is at most 2; we also give a short proof of their result.

1 Introduction

A set of vertices of a graph G is an *odd cycle (vertex) transversal* (or *cover*) if its removal makes G bipartite. An *odd cycle (vertex) packing* in G is a collection of vertex-disjoint odd cycles in G. Let τ and ν respectively denote the minimum size of an odd cycle transversal and the maximum size of an odd cycle packing. Clearly we have $\nu \leq \tau$. Our main result is to show that $\tau \leq 10\nu$ for all planar graphs G. For general graphs, it is known that τ is not bounded by any function of ν. In other words, odd cycles do not satisfy the *Erdös-Pósa property*. In fact, there are graphs known as *Escher walls* [10] for which $\nu = 1$ and τ is arbitrarily large. In [10], Reed proved that the Erdös-Pósa property holds for odd cycles in graphs without Escher wall of height h, for any fixed $h \geq 3$. Since planar graphs do not contain any Escher wall, there exists a function f such that $\tau \leq f(\nu)$ for all planar graphs G. However, the function f implicit in [10] is huge. A similar type of result in this area applies to highly connected graphs G. Thomassen [15] proved that $\tau \leq 2\nu$ for $2^{3^{9\nu}}$-connected graphs. Reed and Rautenbach [9] generalised this to 576ν-connected graphs.

The minimum odd cycle transversal (cover) problem and maximum odd cycle packing problem can be formulated as integer programs whose linear program

⋆ Fellow of the "Fonds de la Recherche Scientifique".
⋆⋆ Supported by NSERC (PGS Master award).
⋆⋆⋆ Canada Research Chair in Graph Theory.
† Supported by NSERC (award 288334-04) and FQRNT (award 2005-NC-98649).

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 35–50, 2005.

relaxations are duals. Letting V denote the vertex set of G and $\mathcal{O}$ denote the set of odd cycles in G, these dual LPs are:

$$\text{Covering LP: } \min \sum_{v \in V} y_v \qquad \text{Packing LP: } \max \sum_{C \in \mathcal{O}} x_C$$

$$\sum_{v:v \in C} y_v \geq 1 \; \forall C \in \mathcal{O} \qquad \sum_{C:v \in C} x_C \leq 1 \; \forall v \in V$$

$$y_v \geq 0 \; \forall v \in V \qquad x_C \geq 0 \; \forall C \in \mathcal{O}$$

Goemans and Williamson [5] gave a constant factor approximation algorithm for the minimum odd cycle transversal problem in planar graphs using the primal-dual method. In doing so they proved that the integrality gap of the covering LP is at most $\frac{9}{4}$. They conjecture that it is actually $\frac{3}{2}$. Our results show that the integrality gap of the packing LP is bounded by a constant. In addition, our structural result generates a polynomial time 10-approximation algorithm for the maximum odd cycle packing problem in planar graphs.

We remark that there are corresponding edge versions of these covering and packing problems. Recently, Král and Voss [7] showed that the minimum size of an odd cycle edge transversal in a planar graph is at most twice the maximum size of an odd cycle edge packing. We also give a short proof of their result.

We conclude this introductory section with an overview of the paper. At the heart of this work lies a connection between odd cycle transversals and T-joins in a certain auxiliary graph. In particular, T will correspond to the set of odd faces of G. In the vertex case the auxiliary graph we need to consider is the face-vertex incidence graph G^+. Specifically, we show that minimum vertex transversals correspond to T-joins in G^+ covering the least number of vertices of G. In the edge case the auxiliary graph is the dual graph G^*. Here minimum transversals correspond to T-joins in G^* with the least number of edges. This relationship was first used by Hadlock [6] to derive a polynomial time algorithm for the maximum cut problem in planar graphs. We present the necessary background on T-joins and T-cuts in Section 2.1 and describe the connection between odd cycle transversals and T-joins in Section 2.2 for the edge case and in Section 2.3 for the vertex case.

In Section 3, we present our proof that a minimum edge transversal has size at most twice the size of a maximum odd cycle edge packing. This proof, as with the proof of Král and Voss, requires the use of the Four Colour Theorem. We use it to show that any laminar 2-packing (defined below) of k odd cycles in G contains $\frac{1}{4}k$ edge-disjoint odd cycles. A result by Lovász on T-joins and T-cuts then guarantees the existence of a 2-packing whose size is twice the minimum size of an odd cycle edge transversal. This gives the result.

To prove our main result we begin by considering three special classes of planar graph. In Section 4.1 we consider graphs in which every pair of odd faces intersect. In Section 4.2 we consider 4-connected graphs such that some (possibly even) face intersects every odd face. Finally in Section 4.3 we examine graphs in which every pair of odd faces is "far" apart. In all of these classes

of graph we show that $\tau \leq 2\nu$. The proof of our main result, given in Section 5, combines the techniques developed for these special cases. We let G be a minimum counterexample to the theorem. We take a minimum collection of faces of G, which we call *centers*, such that every odd face of G intersects some face of the collection. Then we show that G must be 4-connected with all its centers "far" apart. Next, using the techniques of Section 4.2 we find a "local" transversal around each center. At the same time, we find a packing of odd faces around each center. Since the centers are "far" apart the union of these packings is also a packing. We call this union the *local* packing. The results of Section 4.3 allow us to extend the "local" transversals to a transversal of the whole graph. Associated with this transversal we find a different packing of odd cycles, which we call the *global* packing. The size of the transversal we obtain is within a constant factor of the size of the largest of the local packing and the global packing. Our main result then follows.

2 T-Joins and Odd Cycle Transversals

In this section we show how minimum odd cycle edge and vertex transversals of a plane graph G relate to T-joins in the dual graph and face-vertex incidence graph of G, respectively. First, we give some background on T-joins along with two min-max results that we will need.

2.1 Background

Consider any graph H and set of vertices T in H. A *T-join* in H is a set of edges J such that T equals the set of odd degree vertices in the subgraph of H determined by J. There exists a T-join in H if and only if each connected component of H contains an even number of vertices of T. In particular, if H has a T-join then $|T|$ is even. A *T-cut* in H is a cut having an odd number of vertices of T on each side. In other words, whenever a set of vertices X contains an odd number of vertices of T, the cut $\delta(X) = \{xy \in E(H) : x \in X, y \notin X\}$ is a T-cut. The *length* of a T-join is the number of edges it contains. A *packing* of T-cuts is a collection of edge-disjoint T-cuts. Because every T-join intersects every T-cut, the minimum length of a T-join in H is at least the maximum size of a packing of T-cuts in H. In fact, equality holds for bipartite graphs, see Proposition 1 below.

Two sets X and Y of vertices of H are said to be *laminar* if either $X \subseteq Y$ or $Y \subseteq X$ or $X \cap Y = \varnothing$. The sets X and Y are *cross-free* when they are laminar or $X \cup Y = V(H)$. A collection of subsets of $V(H)$ is said to be *laminar* (resp. *cross-free*) if any two of its members are laminar (resp. cross-free). Consider a collection $\mathcal{F}$ of subsets of $V(H)$. Letting $\delta(\mathcal{F}) = \{\delta(X) : X \in \mathcal{F}\}$, the collection of cuts $\delta(\mathcal{F})$ is said to be *laminar* (resp. *cross-free*) whenever $\mathcal{F}$ is.

Proposition 1 (Seymour [13]). *Let H be a bipartite graph and let T be an even set of vertices of H. The minimum length of a T-join in H equals the*

maximum size of a packing of T-cuts in H. The maximum is attained by a cross-free collection of T-cuts. □

The latter proposition implies the next, where a 2-*packing* of T-cuts is a collection of T-cuts such that each edge is contained in at most two T-cuts of the collection.

Proposition 2 (Lovász [8]). *Let H be a graph and T be an even set of vertices of H. The minimum length of a T-join in G equals half the maximum cardinality of a 2-packing of T-cuts in H. The maximum is attained by a cross-free collection of T-cuts.* □

The following observation will be useful in subsequent sections. Its proof is based on standard uncrossing techniques (see, e.g., Proposition 3.4 in [4] or Section 80.7b in [12]) and is not included here due to length restrictions.

Observation 1. *In Propositions 1 and 2, there exists an optimal collection of T-cuts which is laminar and consists only of inclusion-wise minimal T-cuts.*

2.2 Relating Edge Transversals to T-Joins in G^*

Hadlock [6] first noted the following correspondence between odd cycle edge transversals of G and T-joins in its dual graph. Below, G^* denotes the dual graph of G and T the set of odd faces of G, regarded as a subset of $V(G^*)$. We remind the reader that the parity of a face equals the parity of its boundary, counting bridges twice. Note that $|T|$ is always even.

Lemma 1 (Hadlock [6]). *A set of edges F is an odd cycle edge transversal of G if and only if $F^* = \{e^* : e \in F\}$ is a T-join in G^*. Hence, the minimum size of an odd cycle edge transversal of G equals the minimum length of a T-join in G^*.* □

2.3 Relating Vertex Transversals to T-Joins in G^+

We now show how odd vertex cycle transversals of G relate to T-joins in its face-vertex incidence graph. As above, T denotes the set of odd faces of G. The *face-vertex incidence graph* of G is the bipartite graph G^+ on the faces and vertices of G whose edges are the pairs fv, where f is a face of G and v is a vertex of G incident to f. The face-vertex incidence graph is planar because it can be drawn in the plane as follows. Keep all vertices of G as vertices of G^+ and add a new vertex v_f in each face f of G. Then link each new vertex v_f to the vertices of G which are incident to f by an arc whose interior is contained in f. Do this in such a way that two arcs never have a common interior point. The resulting drawing of G^+ is referred to as a *standard drawing*. The relationship between transversals of G and T-joins in the face-vertex incidence graph G^+ is as follows. Below, and henceforth, $F(G)$ denotes the face set of G.

Observation 2. *Let $\delta(X)$ be a T-cut in the face-vertex incidence graph G^+ and let R denote the subgraph of G determined by the edges incident to a face in X and to a face in $\bar{X}$. Then R is Eulerian and has an odd number of edges. Hence, R contains an odd cycle.*

Proof. Pick some vertex v of G. Let $e_1, \ldots, e_d$ denote the edges of G incident to v listed in clockwise order and, for $1 \leq i \leq d$, let f_i be the face of G incident to both e_i and e_{i+1} (we let $e_{d+1} = e_1$). Each face f_i belongs either to X or to $\bar{X}$. Because there is an even number of switches between X and $\bar{X}$ when one goes clockwise around v, the degree of v in R is even. In other words, R is Eulerian. So it can be decomposed into edge-disjoint cycles. Since X contains an odd number of vertices of T, that is, an odd number of odd faces of G, subgraph R has an odd number of edges. □

Lemma 2. *A subset W of $V(G)$ is a transversal of G if and only if the subgraph of the face-vertex incidence graph G^+ induced by $W \cup F(G)$ contains a T-join, that is, every component of the subgraph has an even number of vertices of T.*

Proof. We first prove the forward direction. Suppose, by contradiction, that some connected component X of the subgraph of G^+ induced on $W \cup F(G)$ contains an odd number of vertices of T. Then $\delta(X)$ is a T-cut in G^+. Consider the edges of G incident to a face in X and to a face in $\bar{X}$. These edges determine a subgraph R of G. Let e be an edge of R. None of the endpoints of e belongs to W because otherwise all the faces incident to this endpoint would be in X and e would not belong to R, a contradiction. Therefore, R is vertex-disjoint from W. By Observation 2, we know that R contains an odd cycle. So W is not a transversal, a contradiction.

To prove the backward direction, consider an odd cycle C and a T-join J in G^+ covering some vertices of W and no vertex of $G - W$. Let Y be the set of faces of G contained in C and let $X = Y \cup W$. Because C is odd, X contains an odd number of odd faces, that is, an odd number of elements of T. Because $|T|$ is even, there is an odd number of elements of T in $\bar{X}$ too. It follows that J contains a path P from an element of T in X to an element of T in $\bar{X}$. Let v be any vertex of G on P incident to a face in X and to a face in $\bar{X}$. Then v is a vertex of C covered by J. In other words, W intersects C. Therefore, W is a transversal. □

3 The Edge Case

In this section, we give a short proof that a minimum odd cycle edge transversal has size at most twice the size of a maximum packing of edge-disjoint odd cycles. This result was recently proved by Král and Voss [7]. Their proof is quite long (about 10 pages). Below, we give a concise proof. As with the proof of Král and Voss, our proof relies on the Four Colour Theorem [1, 11].

Theorem 3 (Král and Voss [7]). *The minimum size of an odd cycle edge transversal of G is at most twice the maximum size of an odd cycle edge packing in G.*

Proof. Let τ and ν respectively denote the minimum size of an odd cycle edge transversal of G and the maximum size of an odd cycle edge packing in G. The theorem trivially holds if $\nu = 0$. Assume that $\nu > 0$. By Lemma 1, the minimum size of a T-join in G^* equals τ. By Proposition 2 and Observation 1, there is a laminar family $\mathcal{F}$ of 2τ subsets of $V(G^*)$ such that $\delta(\mathcal{F}) = \{\delta(X) : X \in \mathcal{F}\}$ is a 2-packing of inclusion-wise minimal T-cuts in G^*. Without loss of generality, we can assume that the outer face o of G is odd and that no member of $\mathcal{F}$ contains o.

Let H denote the graph on $\mathcal{F}$ in which X and Y are adjacent whenever the corresponding T-cuts intersect. We claim that H is planar. The claim obviously implies the theorem because, by the Four Colour Theorem, H has a stable set of size at least $|V(H)|/4 = 2\tau/4 = \tau/2$. This implies the desired inequality $\tau \leq 2\nu$. In order to show that H is planar, it suffices to show that every block H' of H is planar. Let $\mathcal{F}'$ denote the vertex set of H'. Since $\mathcal{F}$ is laminar, $\mathcal{F}'$ is also laminar and the set $\mathcal{F}'$ partially ordered by inclusion is a forest, i.e., every point is covered by at most one point. Let X, Y and Z be three distinct elements of $\mathcal{F}'$. The following cannot occur: (i) $X \subseteq Y \subseteq Z$, (ii) $X \subseteq Y$ and $Y \cap Z = \varnothing$. Indeed, if (i) or (ii) holds then every X–Z path in H' intersects Y because $\delta(\mathcal{F})$ is a 2-packing. This contradicts our assumption that H' is a block of H. Then $\mathcal{F}'$ partially ordered by inclusion is either a forest of height 0 (that is, an antichain) or a tree of height 1. In both cases, it is easy to construct a planar drawing for H' from G. Each element of $\mathcal{F}'$ determines a cycle in the plane graph G. In the first case, we pick any point in the bounded face of each of these cycles and connect the points by an arc whenever there is an edge in H' between the two corresponding elements of $\mathcal{F}'$. This can be done in such a way that the resulting graph is planar. The second case is similar. □

4 Special Classes

In this section we show that the minimum size of an odd cycle (vertex) transversal is at most twice the maximum size of an odd cycle (vertex) packing for a collection of special classes of planar graphs. The techniques we develop here will then be applied in the next section to give our main result for general planar graphs. For technical reasons it will be useful to assume that G is signed. A *signed* graph is a graph whose edges are labeled *odd* ('−') or *even* ('+'). In a signed graph, a cycle (or more generally a subgraph) is said to be *odd* if it contains an odd number of odd edges and *even* otherwise. Similarly, a face of a plane signed graph is said to be *odd* if its boundary has an odd number of odd edges, counting bridges twice. Otherwise, the face is said to be *even*. A signed graph is said to be *balanced* if it has no odd cycle. Odd cycle transversals and odd cycle packings are defined as in the unsigned case. Henceforth, in order to

avoid unnecessary repetitions, we abbreviate odd cycle vertex transversal and odd cycle vertex packing respectively as *transversal* and *packing*. We denote by $\tau(G)$ and $\nu(G)$ the minimum size of a transversal of G and the maximum size of a packing in G, respectively. We assume that G has no loops and no multiple edges, with one exception. We allow *odd digons*, i.e., subgraphs with two vertices and two edges between them, one of which is odd and the other even.

4.1 When All Odd Faces Mutually Intersect

The proofs of the following two results could not be included in this version of the article due to length restrictions, but will be included in the journal version. The second result will be used as a base case to prove our main approximate min-max result.

Lemma 3. *Let G be a 3-connected plane graph. Then every face of G is bounded by a cycle. For any two faces f and f' of G whose respective boundaries C and C' intersect, the following holds. Either C and C' share exactly one vertex, or two adjacent vertices and the edge between them.*

Proposition 3. *If every two odd faces of G have intersecting boundaries, then G has a transversal of size at most* 2.

4.2 When Some Face Intersects Every Odd Face

In this section, we consider the graphs that have some face whose boundary intersects the boundary of every odd face.

Proposition 4. *Assume G is 4-connected, simple, has at least five vertices and is such that the boundary of the outer face intersects the boundary of every odd face. Then the minimum size of a transversal of G is at most twice the maximum size of a packing in G.*

Proof. We assume that G is not balanced. Otherwise, the result trivially holds. The hypotheses severely restrict the way face boundaries intersect each other. Consider a vertex y not incident to the outer face. If there are two distinct vertices x, z such that each one of them is incident to the outer face and to a face incident to y, then x and z have to be adjacent. Indeed, there exists a polygon P in $\mathbb{R}^2$ intersecting G exactly in x, y and z. By the Jordan Curve Theorem, we know that all paths from a vertex of G in the bounded region of $\mathbb{R}^2 \setminus P$ to a vertex of G in the unbounded region of $\mathbb{R}^2 \setminus P$ go through x, y or z. If x and z are not adjacent, then the two neighbours of x on the boundary of the outer face lie in a different region of $\mathbb{R}^2 \setminus P$. Hence $X = \{x, y, z\}$ is a cutset of size 3 in G, a contradiction.

Now consider two distinct odd faces f and g different from the outer face. By Lemma 3, the boundaries of faces f and g intersect in a vertex or in a common edge. If the boundaries of f and g share a unique vertex y, then y is incident to the outer face unless the following occurs. There are vertices x and z incident

to the outer face such that xy, xz and yz are edges, x is incident to f and z is incident to g Moreover, x is the only vertex incident to both f and the outer face, and z is the only vertex incident to both g and the outer face. We refer to the triangle on x, y and z as a *junctional triangle*. Note that junctional triangles can be even because G is signed. If the boundaries of f and g intersect in a common edge e then one of the endpoints of e is on the outer face and the other is not. Moreover, in that case f and g cannot both have a common incident edge with the outer face because otherwise $G = K_4$, contradicting the fact that G has at least five vertices.

Enumerate the vertices of the outer face in clockwise order as v_1, v_2, ..., v_n. For the sake of simplicity, let $v_0 = v_n$ and $v_{n+1} = v_1$. Let I be the set of indices i such that there is a junctional triangle containing the edge v_iv_{i+1}. For each $i \in I$, we let u_i be the vertex of the junctional triangle incident to the edge $e = v_iv_{i+1}$ and opposite to e, and we let w_i be any point in the interior of the edge e. For each odd face f different from the outer face, we define an arc A_f contained in the frontier of the outer face, as follows. If the boundary of f intersects the boundary of the outer face in an edge v_iv_{i+1}, then we let A_f be the edge v_iv_{i+1}. Otherwise, the boundary of f intersects the boundary of the outer face in a vertex v_i. If f is incident neither to u_{i-1} nor to u_i then we let A_f be the point $\{v_i\}$. If f is incident to u_{i-1} and not to u_i then we let A_f be the part of the edge $v_{i-1}v_i$ between w_{i-1} and v_i. If f is incident to u_i and not to u_{i-1} then we let A_f be the part of the edge v_iv_{i+1} between v_i and w_i. Finally, if f is incident to both u_i and u_{i-1} then we let A_f be the arc linking w_{i-1} and w_i on the outer face and containing v_i. By construction, two odd faces f and g different from the outer face are incident to some common vertex if and only if their corresponding arcs A_f and A_g have a nonempty intersection.

Let H denote the graph whose vertices are the odd faces different from the outer face and whose edges are the pairs fg such that $A_f \cap A_g \neq \varnothing$. Then H is a circular arc graph. The maximum size of a packing in G is precisely equal to the maximum size of a stable set in H, that is, we have $\nu(G) = \alpha(H)$. We consider the following two cases.

Case 1. There is some point x on the boundary of the outer face that is not in any arc A_f. In this case, H is an interval graph. Let W be a minimum cardinality subset of $\{v_i : 1 \leq i \leq n\} \cup \{w_i : i \in I\}$ meeting all the arcs. By Dilworth's chain partitioning theorem [3], the complement of an interval graph H is perfect, hence we have $|W| = \alpha(H) = \nu(G)$. Now replace each $w_i \in W$ by v_i and v_{i+1}. Let W' be the resulting set of vertices of G. Then W' is a transversal of G of cardinality at most $2|W|$. In other words, we have $\tau(G) \leq 2\nu(G)$.

Case 2. The arcs A_f cover the whole boundary of the outer face. It follows that for each edge $e = v_iv_{i+1}$, the face f_i incident to e and different from the outer face is either an odd face or an even junctional triangle. If f_i is odd, then we let $g_i = f_i$. Otherwise, we let g_i be the odd face incident to v_iu_i. As above, u_i denotes the vertex of the junctional triangle incident to e which is opposite to e. Thus, every edge of the outer face has a corresponding odd face. Note that if the boundaries of g_i and g_j intersect then $i \in \{j-1, j+1\}$ or $i = j$. So if

n is even, then we have $\tau(G) \leq 2\nu(G)$ because $\{v_1, \ldots, v_n\}$ is a transversal of size n and $\{f_1, f_3, \ldots, f_{n-1}\}$ yields a packing of size $n/2$. Now assume that n is odd. Let H' be the graph whose vertices are the faces g_i and whose edges are the pairs $g_i g_j$ such that the boundary of g_i intersects that of g_j and $i \neq j$. By what precedes, we know that the graph H' is a subgraph of the odd cycle with vertex sequence $v_1, \ldots, v_n, v_1$. If H' is not connected, then it has a stable set of size $(n+1)/2$ and we get $\tau(G) \leq 2\nu(G)$ as before. For the rest of the proof, we assume that H' is connected. We claim that either all f_i's are odd faces or all f_i's are even junctional triangles. Otherwise, there is some index i such that f_i is an even junctional triangle and f_{i+1} is an odd face. By what precedes, the boundaries of g_i and g_{i+1} cannot intersect. So our claim holds.

If all f_i's are odd faces then consider vertex v_1. If $\{v_1, \ldots, v_n\} \setminus \{v_1\}$ is a transversal then we have $\tau(G) \leq n-1 \leq 2\nu(G)$ because H' has a stable set of size $(n-1)/2$. Otherwise, there is some odd face f incident to v_1 and to no other v_i. Then $\{f\} \cup \{f_2, f_4, \ldots, f_{n-1}\}$ yields a packing of size $(n+1)/2$. Hence, we have $\tau(G) \leq 2\nu(G)$. If all f_i's are even junctional triangles, then the odd faces of G are exactly the outer face and the faces g_i for $i = 1, \ldots, n$. It is easy to see that $\{u_1, v_3, v_4, \ldots, v_n\}$ is a transversal of size $n-1$. Because H' has a stable set of size $(n-1)/2$, we have $\tau(G) \leq 2\nu(G)$. This concludes the proof. □

We need a slight generalization of Proposition 4. Consider some face f of G, which we refer to as a *center*. The odd faces of G whose boundary intersects the boundary of the center are called the *targets* (around f). In particular, if f is odd then f is itself a target. A *local transversal* is a set W of vertices of G satisfying the following properties:

(i) every target is incident to some vertex of W;
(ii) at most one vertex of W is not incident to the center;
(iii) if $u \in W$ is not incident to the center, then u is incident to exactly two targets.

The proof of Proposition 4 in fact shows:

Lemma 4. *Assume G is 4-connected, simple and has at least five vertices. Let f be a face of G acting as center. Then the minimum size of a local transversal of G is at most twice the maximum number of boundary-disjoint targets in G.* □

4.3 When Odd Faces Are Disjoint

We begin this section by recasting the minimum transversal and the maximum packing problems entirely in terms of T-joins and T-cuts in the face-vertex incidence graph. This slight change of terminology simplifies the proofs and enables us to state our results with more generality. Let H denote any bipartite graph with bipartition $\{A, B\}$, and let T be any even subset of B. The *width* of a T-join in H is the number of vertices of A it covers. The *fringe* of a T-cut $\delta(X)$ in H is the set of vertices of A which have a neighbour in X and a neighbour in $\bar{X}$. Note that the minimum width of a T-join in H is at least the maximum number

of fringe-disjoint T-cuts in H. This is due to the fact that every T-join covers some element in the fringe of every T-cut.

We now relate the above definitions to odd cycle vertex transversal and packing in plane signed graphs. Consider the case where H is the face-vertex incidence graph G^+ of the plane signed graph G, set A is the vertex set of G, set B is the face set of G, and set T is, as before, the set of odd faces of G. By Lemma 2, every T-join in H defines a transversal of G, namely, the vertices of A it covers. Reciprocally, to every transversal W there corresponds a T-join in H which covers some vertices of W and no vertex of $A \setminus W$. So the minimum width of a T-join in H equals the minimum size of a transversal of G. Furthermore, there is a correspondence between T-cuts in H and odd cycles in G. By Observation 2, every T-cut in H determines a Eulerian subgraph of G with an odd number of odd edges. This subgraph contains an odd cycle. The vertex set of the subgraph is the fringe of the T-cut. Reciprocally, every odd cycle in G determines a T-cut in H whose fringe is the vertex set of the cycle. Hence the maximum size of a collection of fringe-disjoint T-cuts in H equals the maximum size of a packing in G. We use the following notation: let ν denote the maximum size of a collection of fringe-disjoint T-cuts in H, let τ denote the minimum width of a T-join in H and let ℓ denote the minimum length of a T-join in H.

Proposition 5. *Let H be any bipartite graph with bipartition $\{A, B\}$ and let T denote any even subset of B. Assume that the shortest path distance $d_H(t, t')$ between any two distinct elements t and t' of T is at least $2c$ for some $c \geq 1$. Then we have*

$$\nu \geq \frac{1}{2}(\ell - |T| + 1) \geq \left(1 - \frac{1}{c}\right)\tau.$$

Proof. Let $\mathcal{F}$ denote a laminar collection of ℓ sets of faces and vertices of G such that $\delta(\mathcal{F}) = \{\delta(X) : X \in \mathcal{F}\}$ is a collection of edge-disjoint T-cuts in H. Such a laminar collection of T-cuts is guaranteed to exist by Proposition 1 and Observation 1.

We claim that whenever X, Y and Z are three distinct elements of $\mathcal{F}$ such that $X \subseteq Y \subseteq Z$ or $X \subseteq Y$ and $Y \cap Z = \varnothing$, then T-cuts $\delta(X)$ and $\delta(Z)$ are fringe-disjoint. It suffices to consider the first case. Suppose there exists an element $a \in A$ which belongs to the fringes of X and Z. In particular, a has a neighbor b in X and a neighbor b' in $\bar{Z}$. If $a \in Y$ then $ab' \in \delta(Y) \cap \delta(Z)$, a contradiction. If $a \in \bar{Y}$ then $ab \in \delta(X) \cap \delta(Y)$, a contradiction. So our claim holds.

The set $\mathcal{F}$ partially ordered by inclusion is a forest. Without loss of generality, we can assume that the leaves of this forest are singletons of the form $\{b\}$ for some $b \in T$. It follows that $\mathcal{F}$ has at most $|T|$ leaves. Note that the claim above implies that two nodes of the forest X and Y are fringe-disjoint unless X is the parent of Y, Y is the parent of X, X and Y are siblings or X and Y are roots.

Rank the children of each node of the forest $\mathcal{F}$ arbitrarily and order its roots arbitrarily also. Let $\mathcal{F}'$ denote the subset of $\mathcal{F}$ formed by all nodes which are ranked first in their respective ordering. Letting λ denote the number of leaves of $\mathcal{F}$, we claim that $\mathcal{F}'$ contains at least $|\mathcal{F}| - \lambda + 1 \geq |\mathcal{F}| - |T| + 1$ elements. Let

X be any node. We define a function $f : \mathcal{F} \to \mathcal{F}$ as follows. If X is a leaf then we set $f(X) = X$. Otherwise, we set $f(X) = f(Y)$, where Y is the first child of X. So $f(X)$ is the "first" leaf amongst the descendants of X. Observe that, for any leaf node Z the preimage of Z, under f, either contains exactly one node in $\mathcal{F}'$ or contains the highest ranked root node. Moreover, by construction, the preimages of any pair of leaves are disjoint. Thus, $\lambda = |\mathcal{F}| - |\mathcal{F}'| + 1$. Our second claim follows.

To obtain a packing of fringe-disjoint T-cuts, colour the elements of $\mathcal{F}'$ black or white in such a way that no parent and child have the same colour, i.e., whenever X is the parent of Y then X and Y have different colours. In other words, colour the subgraph of the Hasse diagram of $\mathcal{F}$ induced on $\mathcal{F}'$ with two colours. Let $\mathcal{F}''$ denote the biggest of the two colour classes. Then $\delta(\mathcal{F}'')$ is a collection of fringe-disjoint T-cuts of size at least $\frac{1}{2}(\ell - |T| + 1)$.

Note that every minimum length T-join in G can be thought of as a perfect matching on T whose edges have become edge-disjoint shortest paths in H. Hence ℓ is at least $\frac{|T|}{2} \cdot 2c$. Note also that τ is at most $\frac{\ell}{2}$ because the width of any T-join is at most half its length. It follows that we have

$$\nu \geq \frac{1}{2}(\ell - |T| + 1) \geq \frac{1}{2}(\ell - |T|) \geq \left(1 - \frac{1}{c}\right)\frac{\ell}{2} \geq \left(1 - \frac{1}{c}\right)\tau. \qquad \square$$

Corollary 1. *If the boundaries of the odd faces of G are pairwise disjoint then the minimum size of a transversal of G is at most twice the maximum size of a packing in G.*

Proof. This follows directly from Proposition 5 with $H = G^+$, $A = V(G)$, $B = F(G)$ and $c = 2$. $\square$

5 Combining the Local and Global Approaches

We are now ready to prove our result for general planar graphs. We will combine the local and global approaches we have described to give our main approximate min-max result. Towards this end, let $\rho(G)$ denote the minimum size of a collection of faces of G such that the boundary of every odd face of G intersects the boundary of some face in the collection. The following two lemmas are simple and we omit their proofs. Lemma 5 implies that $\rho(G') \leq \rho(G)$ for any subgraph G' of G.

Lemma 5. *Let G be a plane signed graph with $\rho(G) = r$, and let $f_1, \ldots, f_r$ be a collection of faces such that for every odd face f there is an index i such that the boundary of f_i intersects the boundary of f. Then we have $\rho(G - e) \leq r$ for each edge e. Moreover, we have $\rho(G - e) \leq r - 1$ if edge e is incident to f_i and f_j for some distinct indices i and j.* $\square$

Lemma 6. *Let G be a plane graph and X be a cutset of G with at most three vertices (we allow the case $X = \varnothing$). If G has no cutset with fewer than $|X|$ elements, then there exists a polygon $P \subset \mathbb{R}^2$ intersecting G only in vertices and such that X is precisely the intersection of P and G and each region of $\mathbb{R}^2 \setminus P$ contains a vertex of G.* □

The next lemma, combined with Lemma 6, will allow us to focus on 4-connected graphs G.

Lemma 7. *Let G be a plane signed graph, let $P \subset \mathbb{R}^2$ be a polygon intersecting G only in vertices, and let $X = P \cap V(G)$. Assume that each region R_1 and R_2 of $\mathbb{R}^2 \setminus P$ contains at least one vertex of G. Then X is a cutset in G. For $i = 1, 2$, let G_i be the part of G contained in the closure of region R_i. Then we have $\rho(G_1) + \rho(G_2) \leq \rho(G) + 2$.*

Proof. Let $\{f_1, \ldots, f_r\}$ denote a collection of $r = \rho(G)$ faces of G such that the boundary of every odd face of G intersects the boundary of some face of the collection. Without loss of generality, we can assume that there are some indices r_1 and r_2 with $r_1 \leq r_2$ such that $f_1, \ldots, f_{r_1}$ are contained in R_1 and incident to no vertex of X, and $f_{r_2}, \ldots, f_r$ are contained in R_2 and incident to no vertex of X. For $i = 1, 2$, let g_i denote the face of G_i containing R_{2-i+1}. Then the boundary of every odd face of G_1 intersects the boundary of some face in $\{f_1, \ldots, f_{r_1}\} \cup \{g_1\}$. Similarly, the boundary of every odd face of G_2 intersects the boundary of some face in $\{f_{r_2}, \ldots, f_r\} \cup \{g_2\}$. The lemma follows. □

Theorem 4. *For every unbalanced plane signed graph G, we have $\tau(G) \leq 7\nu(G) + 3\rho(G) - 8$.*

Proof. Let G be a counterexample with $|V(G)|$ as small as possible, and let $\{f_1, \ldots, f_r\}$ denote any minimum collection of faces of G such that the boundary of every odd face of G intersects the boundary of some face in the collection. Note that we have $r = \rho(G) \geq 1$. We claim: (1) G has a packing of size 2, no transversal of size at most 9, and G is simple; (2) G is 4-connected; (3) the shortest path distance $d_{G^+}(f_i, f_j)$ between f_i and f_j is at least 8 whenever $i \neq j$.

Proof of Claim (1). If G has no packing of size 2, then by Proposition 3, we have

$$\tau(G) \leq 2 = 7 + 3 - 8 \leq 7\nu(G) + 3\rho(G) - 8,$$

a contradiction. So G has a packing of size 2, that is, we have $\nu(G) \geq 2$. Now a similar argument shows that G has no transversal of size at most 9, that is, we have $\tau(G) > 9$. If G is not simple, then it has an odd digon. Letting x and y be the vertices of the digon and $X = \{x, y\}$, we have $\nu(G - X) \leq \nu(G) - 1$. By Lemma 5, we have also $\rho(G - X) \leq \rho(G)$. Therefore, because $G - X$ is unbalanced and has less vertices that G, we have

$$\begin{aligned}\tau(G) &\leq 2 + \tau(G - X)\\ &\leq 2 + 7\nu(G - X) + 3\rho(G - X) - 8\\ &\leq 7\nu(G) + 3\rho(G) + 2 - 7 - 8 \leq 7\nu(G) + 3\rho(G) - 8,\end{aligned}$$

a contradiction. So Claim (1) holds.

Proof of Claim (2). By the previous claim, G has at least 10 vertices. Therefore, to prove the present claim, it suffices to prove that G has no cutset of size 3. However, in order to use Lemma 6 we need to show that G is 3-connected; we leave this straightforward task to the reader. Now assume that G is 3-connected. Suppose that G has a cutset X consisting of three vertices x, y and z. Let $Y = \{y, z\}$. By Lemma 6, there exist induced subgraphs G_1 and G_2 of G and a polygon $P \subset \mathbb{R}^2$ determining two regions R_1 and R_2 in the plane such that P intersects G precisely in x, y and z, and G_i equals the restriction of G to the closure of region R_i, for $i = 1, 2$. It suffices to consider the following two cases. Indeed, if $G_1 - X$ and $G_2 - X$ are both balanced then G has a transversal of size at most 3, contradicting Claim (1).

Case 1. Neither $G_1 - X$ nor $G_2 - X$ is balanced. It follows that neither $G_1 - Y$ nor $G_2 - Y$ is balanced. If we have $\nu(G) \geq \nu(G_1 - Y) + \nu(G_2 - Y)$ then Lemma 7 implies

$$\begin{aligned}\tau(G) &\leq 2 + \tau(G_1 - Y) + \tau(G_2 - Y)\\ &\leq 2 + 7\nu(G_1 - Y) + 3\rho(G_1 - Y) - 8 + 7\nu(G_2 - Y) + 3\rho(G_2 - Y) - 8\\ &\leq 7\nu(G) + 3\rho(G) + 2 + 6 - 16 = 7\nu(G) + 3\rho(G) - 8,\end{aligned}$$

a contradiction. Else, we have $\nu(G) = \nu(G_1 - Y) + \nu(G_2 - Y) - 1$. It follows that every maximum packing of $G_1 - Y$ and every maximum packing of $G_2 - Y$ hit the vertex x. So we have $\nu(G_1 - X) = \nu(G_1 - Y) - 1$, $\nu(G_2 - X) = \nu(G_2 - Y) - 1$ and $\nu(G) = \nu(G_1 - X) + \nu(G_2 - X) + 1$. Therefore, we have

$$\begin{aligned}\tau(G) &\leq 3 + \tau(G_1 - X) + \tau(G_2 - X)\\ &\leq 3 + 7\nu(G_1 - X) + 3\rho(G_1 - X) - 8 + 7\nu(G_2 - X) + 3\rho(G_2 - X) - 8\\ &\leq 7\nu(G) + 3\rho(G) + 3 - 7 + 6 - 16 \leq 7\nu(G) + 3\rho(G) - 8,\end{aligned}$$

a contradiction.

Case 2. $G_1 - X$ is balanced and $G_2 - X$ is not balanced. If G_1 is not balanced, then we have $\nu(G) \geq \nu(G_2 - X) + 1$ and hence

$$\begin{aligned}\tau(G) &\leq 3 + \tau(G_2 - X) \leq 3 + 7\nu(G_2 - X) + 3\rho(G_2 - X) - 8\\ &\leq 7\nu(G) + 3\rho(G) + 3 - 7 - 8 \leq 7\nu(G) + 3\rho(G) - 8,\end{aligned}$$

a contradiction. Otherwise, G_1 is balanced. Consider the graph G_2' obtained from G_2 by adding a triangle on x, y and z to G_2'. We do not add an edge if it is already present in G_2. Since we can easily modify G to get a drawing of G_2', we can regard G_2' as a plane graph. Consider any two distinct vertices u, v in $X = \{x, y, z\}$. Because G_1 is balanced, all u–v paths in G_1 have the same parity. We let the parity of the edge uv in G_2' be the parity of all u–v paths in G_1. Note that we have $\tau(G) \leq \tau(G_2')$ and $\nu(G_2') \leq \nu(G)$. Moreover, we have $\rho(G_2') \leq \rho(G)$, as we now prove. Since G is 3-connected, there is a vertex t in $G_1 - X$ sending three independent paths to x, y and z in G_1. By Lemma 5, if we delete from G all edges which are contained in G_1 except those which belong to one of the three paths, the resulting graph G' satisfies $\rho(G') \leq \rho(G)$. Since

the triangle on x, y, z determines an even face in G'_2, we have $\rho(G'_2) \leq \rho(G')$. Hence, we have $\rho(G'_2) \leq \rho(G)$, as claimed. It follows that we have

$$\tau(G) \leq \tau(G'_2) \leq 7\nu(G'_2) + 3\rho(G'_2) - 8 \leq 7\nu(G) + 3\rho(G) - 8,$$

a contradiction. In conclusion, Claim (2) holds.

Proof of Claim (3). Suppose that $d_{G^+}(f_i, f_j) \leq 6$ for some distinct indices i and j. Let X denote the set of vertices of G on a shortest path between f_i and f_j in G^+. So X contains at most three vertices. By Lemma 5, we have $\rho(G - X) \leq \rho(G) - 1$. Because $G - X$ is not balanced, we have

$$\tau(G) \leq 3 + \tau(G - X) \leq 3 + 7\nu(G - X) + 3\rho(G - X) - 8 \leq 7\nu(G) + 3\rho(G) - 8,$$

a contradiction. So Claim (3) holds.

Now we would like to apply Lemma 4 around each face in the collection $\{f_1, \ldots, f_r\}$. So each face f_i will perform as a center. The targets around f_i are the odd faces of G whose boundary intersects the boundary of f. By Claim (3), whenever g is a target around f_i and g' is a target around f_j with $i \neq j$, the boundaries of g and g' are disjoint. By Lemma 4, for each center f_i there exists a packing of odd cycles $\mathcal{C}_i$ formed by target boundaries, and a local transversal W_i whose size is at most twice the size of packing $\mathcal{C}_i$. Let $\mathcal{C}_{\text{local}}$ denote the union of packings $\mathcal{C}_1, \ldots, \mathcal{C}_r$. Then $\mathcal{C}_{\text{local}}$ is a packing.

Now let $H = G^+$, let $A = V(G)$ and let $B = F(G)$. Consider the graph $\tilde{H}$ obtained from H by contracting, for $1 \leq i \leq r$, all vertices of H at distance at most 2 from f_i to a single vertex $\tilde{f}_i$. Note that $\tilde{H}$ is still bipartite, with bipartition $\{\tilde{A}, \tilde{B}\}$, where

$$\tilde{A} = A \setminus \{a \in A : d_H(a, f_i) \leq 2 \text{ for some } i \text{ with } 1 \leq i \leq r\},$$
$$\tilde{B} = B \setminus \{b \in B : d_H(b, f_i) \leq 2 \text{ for some } i \text{ with } 1 \leq i \leq r\} \cup \{\tilde{f}_i : 1 \leq i \leq r\}.$$

Let $\tilde{T}$ denote the set of those $\tilde{f}_i$'s that correspond to centers f_i which have an odd number of targets around them. So $\tilde{T}$ is an even subset of $\tilde{B}$. Let $\tilde{J}$ denote a minimum length $\tilde{T}$-join in $\tilde{H}$. Then, by Proposition 5, there is a collection of fringe-disjoint $\tilde{T}$-cuts $\delta(\tilde{\mathcal{F}})$ in $\tilde{H}$ such that

$$|\delta(\tilde{\mathcal{F}})| \geq \frac{1}{2}(|\tilde{J}| - |\tilde{T}| + 1) \geq \frac{1}{2}(|\tilde{J}| - r + 1) \Rightarrow |\tilde{J}| \leq 2|\delta(\tilde{\mathcal{F}})| + r - 1.$$

This collection of fringe-disjoint $\tilde{T}$-cuts yields a packing of odd cycles $\mathcal{C}_{\text{global}}$ in G, of the same size. The $\tilde{T}$-join $\tilde{J}$ defines a set of edges J_{global} in $H = G^+$, as follows. Every edge of $\tilde{J}$ that belongs to H is kept as it is. Every other edge of $\tilde{J}$ is of the form $v\tilde{f}_i$ and is replaced by any shortest path between v and f_i in H. Because we have $d_{\tilde{H}}(\tilde{f}_i, \tilde{f}_j) \geq 4$ whenever $i \neq j$ and because $\tilde{J}$ is the edge-disjoint union of shortest paths between pairs of vertices of $\tilde{T}$, the length of J_{global} is at most twice the length of $\tilde{J}$.

For each local transversal W_i, let J_i denote the set of edges vf of the face-vertex incidence graph G^+ such that $v \in W_i$ and f is a target around f_i incident

to v. Let J_{local} denote the union of $J_1, \ldots, J_r$. The union of J_{local} and J_{global} contains a T-join, say J. Because the width of J is at most the width of J_{local} plus the width of J_{global} and because the width of a T-join is at most half of its length, the width of J is at most

$$\sum_{i=1}^{r} |W_i| + \frac{1}{2}|J_{\text{global}}| \leq 2|\mathcal{C}_{\text{local}}| + 2|\mathcal{C}_{\text{global}}| + r - 1 \leq 4\nu(G) + \rho(G) - 1.$$

By Claim (1), we have $\nu(G) \geq 2$. Therefore, we have

$$\tau(G) \leq 4\nu(G) + \rho(G) - 1 \leq 7\nu(G) + 3\rho(G) - 8,$$

a contradiction. This concludes the proof of the theorem. □

Because $\rho(G)$ is at most the size of any inclusion-wise maximal collection of boundary-disjoint odd faces in G, which is in turn at most $\nu(G)$, we obtain our main result from Theorem 4.

Corollary 2. *For every plane signed graph G, we have $\tau(G) \leq 10\nu(G)$.* □

References

1. K. Appel and W. Haken, "A proof of the four color theorem", *Discrete Mathematics*, **16**, pp179-180, 1976.
2. R. Diestel, *Graph Theory (Second Edition)*, Number 173 in Graduate Texts in Mathematics. Springer-Verlag, New-York, 2000.
3. R.P. Dilworth, "A decomposition theorem for partially ordered sets", *Annals of Mathematics*, **51**, pp161-166, 1950.
4. A. Frank, "A survey on T-joins, T-cuts, and conservative weightings", In *Combinatorics, Paul Erdös is eighty, Vol. 2 (Keszthely, 1993)*, volume 2 of *Bolyai Society Mathematical Studies*, pp213-252. János Bolyai Mathematical Society, Budapest, 1996.
5. M.X. Goemans and D.P. Williamson, "Primal-dual approximation algorithms for feedback problems in planar graphs", *Combinatorica*, **18**, pp37–59, 1998.
6. F. Hadlock. "Finding a maximum cut of a planar graph in polynomial time", *SIAM Journal on Computing*, **4**, pp221–225, 1975.
7. D. Král and H. Voss, "Edge-disjoint odd cycles in planar graphs", *Journal of Combinatorial Theory, Series B*, **90**, pp107-120, 2004.
8. L. Lovász, "2-matchings and 2-covers of hypergraphs", *Acta Mathematica Academiae Scientiarum Hungaricae*, **26**, pp433-444, 1975.
9. D. Rautenbach and B. Reed, "The Erdős-Pósa property for odd cycles in highly connected graphs", *Combinatorica*, **21**, pp267-278, 2001. Paul Erdős and his mathematics (Budapest, 1999).
10. B. Reed, "On mangoes and blueberries", *Combinatorica*, **19**, pp267-296, 1999.
11. N. Robertson, D. Sanders, P. Seymour, and R. Thomas, "The four colour theorem", *Journal of Combinatorial Theory, Series B*, **70**, pp2-44, 1997.
12. A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, Number 24 in Algorithms and Combinatorics. Springer Verlag, 2003.

13. P. Seymour, "On odd cuts and plane multicommodity flows", *Proceedings of the London Mathematical Society (Third Series)*, **42**, pp178-192, 1981.
14. P. Seymour, "Packing directed circuits fractionally", *Combinatorica*, **15**, pp281-288, 1995.
15. C. Thomassen, "On the presence of disjoint subgraphs of a specified type", *Journal of Graph Theory*, **12**, pp101-111, 1988.

Disjoint Cycles: Integrality Gap, Hardness, and Approximation

Mohammad R. Salavatipour[1,⋆] and Jacques Verstraete[2,⋆⋆]

[1] Department of Computing Science,
University of Alberta,
Edmonton, AB, Canada T6G 2E8
mreza@cs.ualberta.ca
[2] Department of Combinatorics and Optimization,
University of Waterloo,
Waterloo, ON, Canada N2L 3G1
jverstra@math.uwaterloo.ca

Abstract. In the edge-disjoint cycle packing problem we are given a graph G and we have to find a largest set of edge-disjoint cycles in G. The problem of packing vertex-disjoint cycles in G is defined similarly. The best approximation algorithms for edge-disjoint cycle packing are due to Krivelevich et al. [16], where they give an $O(\sqrt{\log\ })$-approximation for undirected graphs and an $O(\sqrt{\ })$-approximation for directed graphs. They also conjecture that the problem in directed case has an integrality gap of $(\sqrt{\ })$. No non-trivial lower bound is known for the integrality gap of this problem. Here we show that both problems of packing edge-disjoint and packing vertex-disjoint cycles in a directed graph have an integrality gap of $(\frac{\log n}{\log\log n})$. This is the first super constant lower bound for the integrality gap of these problems. We also prove that both problems are quasi-NP-hard to approximate within a factor of $(\log^{1-\epsilon}\ )$, for any $\ > 0$. For the problem of packing vertex-disjoint cycles, we give the first approximation algorithms with ratios $O(\log\ )$ (for undirected graphs) and $O(\sqrt{\ })$ (for directed graphs). Our algorithms work for the more general case where we have a capacity c_v on every vertex and we are seeking a largest set $\mathcal{C}$ of cycles such that at most c_v cycles of $\mathcal{C}$ contain .

1 Introduction

We study approximation algorithms, lower and upper bounds for the integrality gaps, and hardness results for the problems of packing disjoint cycles in a graph (directed or undirected). In the problem of packing edge-disjoint cycles (EDC), we are given a graph G (which can be directed or undirected) and we have to find

⋆ Supported by a faculty startup grant, Department of Computing Science, University of Alberta.
⋆⋆ Supported by NSERC.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 51–65, 2005.

a largest set of edge-disjoint cycles in G. The problem of packing vertex-disjoint cycles (VDC) is defined similarly: find the maximum number of vertex-disjoint cycles in G. The EDC problem has been studied extensively both in undirected and directed graphs (see [16, 4, 6, 21]). Both EDC and VDC are fundamental problems in graph theory with applications in several areas (see the discussion in [4] for an application in computational biology and reconstruction of evolutionary trees). Let's denote by $\nu_e(G)$ and $\nu_v(G)$ the sizes of largest collections of edge-disjoint and vertex-disjoint cycles in G, respectively. It is well known that computing $\nu_e(G)$ and $\nu_v(G)$ are NP-hard even for undirected graphs. This motivates the study of approximation algorithms for these problems. An algorithm is called an α-approximation for a maximization problem if the solution returned by the algorithm is at least a factor $1/\alpha$ of the optimal solution, and α is called the approximation (or performance) ratio of the algorithm. A natural generalization of EDC is the problem of S-cycle packing (denote by s-EDC). In this problem, along with G we are given a subset S of vertices of G and the goal is to find maximum number of edge-disjoint S-cycles in G, i.e. cycles each of which contains a vertex of S. The analogous problem of s-VDC is defined as finding maximum number of cycles in G each of which contains a vertex of S and are disjoint on the vertices in $V - S$.

Known Results: Carpara et al. [6] showed that a simple greedy algorithm yields an $O(\log n)$-approximation for computing $\nu_e(G)$ and that the problem is APX-hard even for planar graphs (i.e. for an absolute constant $\epsilon_0 > 0$, no $(1 + \epsilon_0)$-approximation exists unless P=NP). Very recently, Krivelevich et al. [16] showed that the greedy algorithm of [6] actually yields an $O(\sqrt{\log n})$-approximation for EDC and gave examples to show that this is tight. In fact they proved an upper bound of $O(\sqrt{\log n})$ for the integrality gap of EDC . For directed graphs, they gave an $O(\sqrt{n})$-approximation for EDC and an $O(n^{\frac{2}{3}})$-approximation for s-EDC . Their algorithm for directed EDC shows that the integrality gap of EDC in directed graphs is at most $O(\sqrt{n})$. No non-trivial lower bounds for the integrality gap of EDC in directed setting is known. The authors in [16] conjectured that there is a lower bound of $\Omega(\sqrt{n})$ for this integrality gap.

Related Results: The dual problems of packing cycles (known as feedback sets problems) are also very well studied problems in both directed and undirected settings. The dual problem of VDC , known as Feedback Vertex Set (FVS), is the problem of finding minimum number of vertices in a graph whose removal makes the graph acyclic. This problem and its generalization (in which every vertex has a weight and we seek to minimize the total weight of selected vertices) has 2-approximation algorithms in undirected graphs (see [3, 5, 10]). The problem of finding minimum number of edges in a graph that meet every cycle (FES) is trivial for undirected graphs (complement of a spanning tree). For directed graphs, there is an easy reduction from FES to FVS. Seymour [21]

showed that, if the optimal fractional FVS in a directed graph G has value φ^* then the optimal integral FVS in G has value at most $O(\varphi^* \log \varphi^* \log\log \varphi^*)$. This yields an $O(\log \varphi^* \log\log \varphi^*)$-approximation algorithm for FVS in directed graphs [13]. Alon and Seymour (see [21]) showed that the integrality gap of FVS is $\Omega(\log \varphi^*)$.

As far as we know, the only hardness result for the problems of packing disjoint cycles is the the APX-hardness result proved in [6] for undirected EDC , and no better hardness or even a super constant lower bound for the integrality gap of EDC in directed graphs is known. As mentioned earlier, Krivelevich et al. [16] have conjectured that this gap is $\Omega(\sqrt{n})$. This conjecture seems conceivable, given the fact that the well known similar problem of edge-disjoint paths (EDP) has such an integrality gap (even in undirected case). Here we take the first step toward tightening this gap. Our main result is that EDC in directed setting has an integrality gap of $\Omega(\frac{\log n}{\log\log n})$. More importantly, we prove that it is quasi-NP-hard to approximate $\nu_e(G)$ within a factor of $\Omega(\log^{1-\epsilon} n)$ for any $\epsilon > 0$. Under stronger complexity assumptions we can prove a slightly better hardness of $\Omega(\frac{\log n}{\log\log n})$. As we will see, there are quite easy reductions between EDC and VDC in directed graphs. Therefore, our hardness result for EDC carries over to VDC , i.e. VDC in directed graphs is hard to approximate within a factor of $\Omega(\log^{1-\epsilon} n)$.

We also consider the capcitated version of cycle packing problems. In the capacitated version of VDC , we are given a graph $G(V, E)$ with a positive capacity c_v on every vertex $v \in V$ and the goal is to find a largest collection $\mathcal{C}$ of cycles in G such that each vertex v belongs to at most c_v cycles of $\mathcal{C}$. We abuse the notation slightly by calling this capacitated version VDC_c , although the cycles are not required to be vertex-disjoint anymore. For undirected VDC_c , we give an $O(\log n)$-approximation. This also shows an upper bound of $O(\log n)$ on the integrality gap of VDC_c in undirected graphs. We are not aware of any earlier approximation algorithm for this problem. For directed VDC_c , we show that a simple randomized rounding algorithm (similar to the ones we presented in [9]) yields an $O(\sqrt{n})$-approximation. The same algorithm works for the even more general case of capacitated (directed) s-VDC , where every vertex $v \in V - S$ has a capacity c_v and we have to find a largest collection $\mathcal{C}$ of S-cycles (i.e. cycles that intersect S) such that each vertex $v \in V - S$ belongs to at most c_v cycles. We denote this problem by s-VDC_c . Note that for the s-EDC [16] gave an $O(n^{\frac{2}{3}})$ greedy approximation algorithm. It can be shown (as we will see) that this upper bound is tight for their greedy algorithm.

Remark: For the capacited version of EDC (denoted by EDC_c), as noted by the authors in [16], their results for EDC can be easily extended to an $O(\sqrt{\log n})$-approximation algorithm for undirected EDC_c and an $O(\sqrt{n})$-approximation for directed EDC_c .

Throughout the paper, we use n to denote the number of vertices of the input graph.

2 Approximation Algorithms

Let $G(V, E)$ be the given graph with a capacity c_v for every vertex $v \in V$, and let $\mathcal{C}(G)$ denote the set of all cycles of G. The following is the standard integer program (IP) formulation of VDC$_c$:

$$\begin{array}{ll} \text{maximize} & \sum_{C\in\mathcal{C}} x_C \\ \text{subject to} & \forall v \in V : \sum_{C:v\in C} x_C \le c_v \\ & \forall C \in \mathcal{C} : \quad x_C \in \{0,1\} \end{array} \tag{1}$$

Trivially, for VDC all c_v values are equal to 1. The fractional VDC$_c$ problem is the linear program relaxation of this IP. The integer program formulation for EDC$_c$ is defined similarly. Here, we will have a constraint of the form $\sum_{C:e\in C} x_C \le c_e$, for every edge $e \in E$. For the EDC problem, all the capacities are 1. Clearly, the solutions to the fractional problems (LP's) are upper bounds to the solutions for the corresponding integral problems (IP's). Let $\nu_v^*(G)$ and $\nu_e^*(G)$ denote the values of optimal fractional solutions for VDC and EDC , respectively.

Theorem 1. *There is an $O(\log n)$-approximation for (undirected) VDC$_c$.*

Proof. We are going to incorporate a technique from [18] with the simple greedy algorithm, similar to the one for the EDC in [6, 16]. The idea of this greedy algorithm was implicit in [11].

Consider the problem of fractional VDC$_c$, i.e. the LP corresponding to the IP given in (1). Note that the separation oracle for the dual of this LP is the problem of finding a shortest cycle in a weighted graph. Since this problem can be solved in polynomial time, using the same method as in Theorem 4.1 of [15] (or [7]) we can solve the primal LP in polynomial time. Let $\varphi(G)$ and $\varphi^*(G)$ denote the values of optimal solution to the IP and the corresponding LP for VDC$_c$. Let $Y = \{x_1, \ldots, x_p\}$ be the set of primal variables that have value > 0 in the optimal fractional solution. One of the features of the algorithm of [15] is that p (the number of fractional cycles) is polynomial in n, even-though the LP has an exponential number of variables. If $\sum_{i=1}^{p} \lfloor x_i \rfloor \ge \frac{1}{\log n} \sum_{i=1}^{p} x_i$ then $Y' = \{\lfloor x_1 \rfloor, \ldots, \lfloor x_p \rfloor\}$ is an integral solution with value at least $\frac{\varphi^*(G)}{\log n}$, which is at least $\frac{\varphi(G)}{\log n}$. In this case the algorithm returns the cycles corresponding to the variables in Y' and stops. Otherwise, if $\sum_{i=1}^{p} \lfloor x_i \rfloor < \frac{1}{\log n} \sum_{i=1}^{p} x_i$ then

$$\varphi^*(G) = \sum_{i=1}^{p} x_i = \sum_{i=1}^{p} \lfloor x_i \rfloor + \sum_{i=1}^{p} (x_i - \lfloor x_i \rfloor) < \frac{\varphi^*(G)}{\log n} + p.$$

Therefore, with $Q = p \cdot (\frac{\log n}{\log n - 1})$: $\varphi^*(G) < Q$. This implies that for every vertex $v \in G$, at most a value of $\min\{c_v, Q\}$ of capacity of v is used in any optimal (fractional or integral) solution. So we can decrease the capacity c_v of every vertex v to $\min\{c_v, Q\}$.

We are also going to assign capacities to the edges of G. Initially, every edge has infinite capacity. Throughout the algorithm, we will replace a vertex v of degree 2 with neighbors u and w with an edge between u and w with capacity c_v. We perform the following algorithm on G as long as G has a cycle. The algorithm is based on the greedy algorithm proposed in [6] for the EDC . Initially $\mathcal{C} = \emptyset$.

1. While G contains a vertex v of degree ≤ 1 or with capacity 0 delete v (and all the edges incident with it).
2. While G contains a vertex v of degree 2 with neighbors u and w delete v and add an edge uw with capacity c_v to G.
3. Find a shortest cycle C in G and add it to $\mathcal{C}$, decrease the capacity of every vertex in C and every edge with finite capacity in C by 1. Go to step 1.

It is easy to see that steps 1 and 2 don't change the value of an optimal solution. Also, since the capacity of every vertex is polynomial in n, the size of the graph is always a polynomial factor of the initial size of G. Let S_i be the i'th iteration in which we perform step 3 and let G_i be the graph at the beginning of iteration S_i, and $n_i = |G_i|$. It is well known that every graph with minimum degree at least 3 on n vertices has girth (size of the shortest cycle) at most $O(\log n)$ [11]. Since graph G_i has minimum degree 3, the girth of G_i is at most $O(\log n_i)$. Therefore, the cycle found in step S_i intersects at most $O(\log n_i)$ (which is $O(\log n)$) cycles of the optimal fractional solution. This is true for every step S_i. Thus, $\varphi^*(G) \leq O(\log n)|\mathcal{C}|$, i.e. the algorithm is an $O(\log n)$ approximation. □

Our next theorem gives an $O(\sqrt{n})$-approximation algorithm for directed s-VDC$_c$. First we show that the greedy algorithm of [16] for s-EDC (and its adapted version for s-VDC) will have a ratio of at least $\Omega(n^{\frac{2}{3}})$. We give the construction of a graph G which is a simple modification of a construction given by Chekuri and Khanna [8] for the problem of edge-disjoint paths (EDP) in directed graphs. G consists of two layered graphs G_1 and G_2, where G_1 contains layers $X_1, \ldots, X_q$ and G_2 contains layers $Y_1, \ldots, Y_q$ of vertices, with $q = n^{\frac{2}{3}}$. Each X_i and Y_i has $n^{\frac{1}{3}}$ vertices and every vertex in X_i (in Y_i), $1 \leq i < q$, is connected by an edge to every vertex in X_{i+1} (in Y_{i+1}). For every odd value of $i \leq q-2$ pick a representative vertex x_i^r from X_i and one y_{i+1}^r from Y_{i+1}. Connect x_i^r to y_{i+1}^r and connect y_{i+1}^r to x_{i+2}^r. Finlay put an edge between every vertex in Y_q to every vertex in X_1. Let $S = X_1$. It is easy to verify that G has $\Omega(n^{\frac{2}{3}})$ edge-disjoint S-cycles (and they are in fact vertex disjoint on $V - S$). If the greedy algorithm picks in its first S-cycle all the edges between G_1 and G_2 then it finds only one cycle (since the edges from G_1 to G_2 form a cut of size $O(q)$ from G_1 to G_2). Therefore, the ratio of greedy algorithm proposed in [16] for s-EDC (and its analogous for s-VDC) is at least $\Omega(n^{\frac{2}{3}})$.

Theorem 2. *There is an $O(\sqrt{n})$-approximation algorithm for directed s-VDC$_c$.*

Proof. The proof of this theorem follows the same steps as the approximation algorithms of [9] for disjoint Steiner trees in directed graphs. Let $G(V, E)$, set

$S \subseteq V$, and capacity c_v for every vertex $v \in V - S$ be the given instance of s-VDC$_c$. Consider the IP/LP formulation of the s-VDC$_c$ problem, which is the same as (1) given for VDC$_c$ except that $\mathcal{C}(G)$ will be the set of all S-cycles of G. Consider the optimal fractional solution which is again computed using the technique of [15] or [7]. We can compute this fractional solution in polynomial time since the separation oracle for the dual LP is the problem of finding a shortest S-cycle in a weighted graph, and this can be computed in polynomial time. Consider a solution to the LP of s-VDC$_c$ and let $X^* = \{x_1^*, x_2^*, \dots x_{p(n)}^*\}$ be the set of fractional cycles in this optimal fractional solution, where $p(n)$ is some polynomial in n. For every S-cycle C with fractional value $x_C^* \geq 1$ we "take out" $\lfloor x_C^* \rfloor$ copies of that cycle from the graph and put them in the final integral solution that we are computing. This way we will find a set $\mathcal{C}_1$ of size at least $\varphi_1^* = \sum_{i=1}^{p(n)} \lfloor x_i^* \rfloor$ integral S-cycles. We also decrease the capacity of every vertex in cycle C by $\lfloor x_C^* \rfloor$, and replace x_C^* with $x_C^* - \lfloor x_C^* \rfloor$ in the fractional solution. Now let $\varphi_2^* = \sum_{i=1}^{p(n)} x_i^*$. Note that the value of the solution to the LP of s-VDC$_c$ is $\varphi_1^* + \varphi_2^*$. We show how to find a set $\mathcal{C}_2$ of size at least $O(\frac{\varphi_2^*}{\sqrt{n}})$ integral cycles. The final solution will be $\mathcal{C}_1 \cup \mathcal{C}_2$ which clearly has size at least $O(\varphi_1^* + \frac{\varphi_2^*}{\sqrt{n}})$, and since the value of the solution to the original LP is $\varphi_1^* + \varphi_2^*$, we get an $O(\sqrt{n})$-approximation.

If $\varphi_2^* \leq 30\sqrt{n}$ then it is enough to find just one S-cycle in G and place it in $\mathcal{C}_2$. So let's assume that $\varphi_2^* > 30\sqrt{n}$. For every cycle $C \in \mathcal{C}(G)$, pick that cycle with probability x_c^*/λ for a $\lambda > 0$ to be defined soon. Define Y_C to be random variable that is 1 if and only if cycle C is selected. So for $Y = \sum_{C \in \mathcal{C}(G)} Y_C$ (i.e. total number of cycles placed in $\mathcal{C}_2$), we have:

$$E[Y] = \sum_{C \in \mathcal{C}(G)} Pr[Y_C = 1] = \sum_{C \in \mathcal{C}(G)} \frac{x_C^*}{\lambda} = \frac{\varphi_2^*}{\lambda}.$$

Define the bad event A_v to be the event that more than c_v cycles containing vertex $v \in V - S$ are selected. We can show that with positive probability none of these events happens (so no vertex capacity is violated) *and* that the number of cycles selected is at least $O(\varphi_2^*/\lambda)$. We borrow the following lemma from [9]:

Lemma 1. *[9] Assume that $A = \{a_1, \dots, a_n\}$ is a set of n non-negative reals and let $\mathcal{A}_k$ be the set of all subsets of size k of A. If $\sum_{i=1}^n a_i \leq Q$, then $\sum_{\{a_{i_1}, \dots, a_{i_k}\} \in A_k} a_{i_1} a_{i_2} \dots a_{i_k} \leq \binom{n}{k} (Q/n)^k$.*

For every vertex $v \in V - S$, denote the number of fractional cycles C with $x_C^* > 0$ that contain v by ψ_v. By this definition:

$$\Pr[A_v] \leq \sum \prod_{i=1}^{c_v+1} x_{C_{a_i}}^*/\lambda,$$

where the summation is over all subsets $\{C_{a_1}, \ldots, C_{a_{c_v+1}}\}$ of size c_v+1 of cycles with $x^*_{C_{a_i}} > 0$ that contain vertex v. Therefore, using Lemma 1:

$$\Pr[A_v] \leq \binom{\psi_v}{c_v+1}\left(\frac{c_v}{\lambda\psi_v}\right)^{c_v+1} \leq \left(\frac{v\psi_v}{c_v+1}\right)^{c_v+1}\left(\frac{c_v}{\lambda\psi_v}\right)^{c_v+1} \leq \frac{e^2}{\lambda^2},$$

where we have used the fact $\binom{n}{k} \leq (\frac{en}{k})^k$ for the second inequality. It is intuitively clear that if $\overline{A_v}$ holds then it does not increase the probability of any other $A_{v'}$. In other words, events $\overline{A_v}$ are "positively correlated". Therefore: $\Pr[\bigwedge_{v\in V-S}\overline{A_v}] \geq \prod_{v\in V-S}\Pr[\overline{A}_v] \geq (1-\frac{e^2}{\lambda^2})^n$. Also, by Chernoff bound, for $0 \leq \delta < 1$: $\Pr[Y < (1-\delta)\mathrm{E}[Y]] \leq e^{-\delta^2\varphi_2^*/2\lambda}$. Thus:

$$\Pr[(Y < (1-\delta)\mathrm{E}[Y]) \vee (\exists v \in V-S : A_v)] \leq e^{-\delta^2\varphi_2^*/2\lambda} + 1 - (1-e^2/\lambda^2)^n.$$

If we show that for suitable δ and λ: $(1-e^2/\lambda^2)^n > e^{-\delta^2\varphi_2^*/2\lambda}$ then using the method of conditional probability, we can efficiently find a selection $\mathcal{C}_2$ of S-cycles such that $|\mathcal{C}_2| \geq (1-\delta)\varphi_2^*/\lambda$ and that no vertex capacity constraint is violated. If $\varphi_2^* \leq n$ then with $\delta = \frac{1}{2}$ and $\lambda = e\sqrt{n}$ we find a collection $\mathcal{C}_2$ of S-cycles that obey the capacity constraints of vertices with $|\mathcal{C}_2| \geq \varphi_2^*/2e\sqrt{n}$. If $\varphi_2^* > n$ then there is a constant $\sigma > 0$ such that with $\delta = \frac{1}{2}$ and $\lambda = \sigma$: $(1-e^2/\lambda^2)^n > e^{-\delta^2\varphi_2^*/2\lambda}$. Again, we can find a collection $\mathcal{C}_2$ of S-cycles that satisfy the capacity constraints of vertices and $|\mathcal{C}_2| \geq \frac{\varphi_2^*}{2\sigma}$. In any case, we find a set $\mathcal{C}_2$ of size at least $\Omega(\frac{\varphi_2^*}{\sqrt{n}})$. Therefore, the algorithm is an $O(\sqrt{n})$-approximation for s-VDC$_c$. □

3 Integrality Gap and Hardness of Directed EDC and VDC

In this section, we prove that each of EDC and VDC has an integrality gap of $\Omega(\frac{\log n}{\log\log n})$. Furthermore, each of $\nu_v(G)$ and $\nu_e(G)$ is quasi-NP-hard to approximate within a factor of $O(\log^{1-\epsilon} n)$, for any $\epsilon > 0$. First, we provide approximate preserving reductions between EDC and VDC .

Theorem 3. *Given a directed graph $G(V,E)$ as an instance of VDC (of EDC) there is an instance $G'(V',E')$ of EDC (of VDC) with $|G'| = poly(|G|)$, such that G has k vertex-disjoint cycles (edge-disjoint cycles) if and only if G' has k edge-disjoint cycles (vertex-disjoint cycles).*

Proof. 1st direction: For each node $v \in V$, G' contains two nodes v_1, v_2. We add v_1v_2 to E'. Furthermore, for every edge $uv \in E$ we create an edge u_2v_1 in E' and for every edge $vw \in E$ we create an edge v_2w_1 in E'. It is easy to see that if $\mathcal{C}$ is a collection of integral (or fractional) vertex-disjoint cycles in G with size k then there is a collection $\mathcal{C}'$ of k integral (or fractional) edge-disjoint cycles in G'. Conversely, suppose that $\mathcal{C}'$ is a collection of k integral (or fractional) edge-disjoint cycles in G'. Then for every edge v_1v_2 (corresponding to

a vertex $v \in V(G)$) there is at most a total of one integral (or fractional) cycle(s) containing that edge. Therefore, by contracting the edges of the form v_1v_2 on each cycle of $\mathcal{C}'$ we obtain a collection of k integral (or fractional) vertex-disjoint cycles in G.

2nd direction: Suppose G is an instance of EDC . For every edge xy in G create a vertex v_{xy} in G'. For every vertex $x \in G$ with ingoing edges $y_1x, y_2x, \ldots, y_px$ and outgoing edges $xz_1, xz_2, \ldots, xz_q$ add the following edges to G': $v_{y_ix}v_{xz_j}$ for every $1 \leq i \leq p$ and $1 \leq j \leq q$. It can be seen that G has k edge-disjoint integral (or fractional) cycles if and only if if G' has k vertex-disjoint integral (or fractional) cycles. □

So it is enough to prove our lower bound for integrality gap and the hardness result for EDC . Then we use Theorem 3 to deduce a similar result for VDC .

Theorem 4. *The directed EDC problem has an integrality gap of* $\Omega(\frac{\log n}{\log\log n})$.

We give the construction of a graph G on n vertices, such that $\frac{\nu_e^*(G)}{\nu_e(G)} \in \Omega(\frac{\log n}{\log\log n})$. Our starting point is a grid-like graph which gives the $\Omega(\sqrt{n})$ integrality gap for the well-known problems of disjoint paths. An instance of the edge-disjoint paths (EDP) problem consists of a (directed) graph G with pairs of vertices s_i, t_i, for $1 \leq i \leq k$, and the goal is to connect maximum number of pairs s_i, t_i using edge-disjoint paths. The vertex-disjoint paths (VDP) problem is defined similarly.

Let r be a positive integer and define a directed graph which consists of vertices s_i, t_i $(1 \leq i \leq r)$ together with vertices h_{ij}, u_{ij}, v_{ij}, $1 \leq j \leq i \leq r$. There is an edge from h_{ij} to u_{ij} and an edge from h_{ij} to v_{ij} $(1 \leq j \leq i \leq r)$. There are also edges $u_{ij}h_{i(j+1)}$ and $v_{ij}h_{(i+1)j}$ for $1 \leq j < i < r$. Furthermore, for every $1 \leq i < r$ it has edges $u_{ii}h_{(i+1)(i+1)}$, s_ih_{i1}, and $v_{ri}t_i$. Finally $u_{r(r-1)}$ is connected to t_r. Since this graph has a drawing on the plane, there cannot be two *vertex-disjoint* paths P_i and P_j $(1 \leq i \neq j \leq r)$ where P_i starts from s_i and ends in t_i and P_j starts from s_j and ends in t_j. Because we want to have edge-disjoint property, we "split" every vertex h_{ij} into two copies h_{ij}^1 and h_{ij}^2, where the ingoing edges of h_{ij} are now going into h_{ij}^1 and the outgoing edges of h_{ij} are

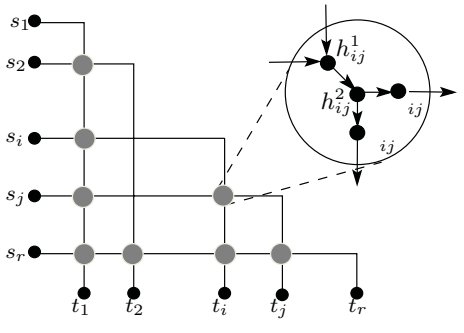


Fig. 1. Construction of D_r: each gray circle corresponds to an intersection module

going out of h^2_{ij} and put the edge $h^1_{ij}h^2_{ij}$ in (see Figure 1). Let's call this graph D_r and the subgraph induced by four vertices $h^1_{ij}, h^2_{ij}, u_{ij}, v_{ij}$ an *intersection module* of D_r. Again, it is easy to see that there cannot be two *edge-disjoint* paths from s_i's to t_i's (because we can route at most one path through every intersection module). Note that:

Fact 1: The half-integral fractional solution for EDP in D_r has value $\geq \frac{r}{2}$.

This creates a gap of $\Omega(r)$, which is $\Omega(\sqrt{n})$, with n being the number of vertices in the graph. We will use this fact again, later on. A natural attempt to extend this result to the cycle packing problem would be to add directed edges t_is_i, for $1 \leq i \leq r$. Unfortunately, this new graph will have an integral solution of value $\Omega(r)$ (for e.g. consider the directed cycle that goes from s_1 to t_r along the diagonal path, then to s_r and to t_1 and back to s_1. We can pick $\frac{r}{4}$ such cycles). So this doesn't create the desired gap. The problem here is caused because the cycles are not bond to follow a path directly from s_i to t_i (they may go through other s_j's and t_j's before reaching t_i). Our idea to resolve this problem is to make it "too costly" for the cycles to do so. In other words, we are going to combine many copies of D_r in a special manner so that if a cycle consist of a "non-trivial" path from s_i to t_i then it has a very long length; so long that we cannot have many of them. This will create the desired gap.

Using two copies of graph D_r we construct another graph H_r in the following way. Consider graph D_r with input vertices $s_1, \ldots, s_r$ and output vertices $t_1, \ldots, t_r$. Take two copies of this graph, D^1_r and D^2_r, and identify (only) the input vertices of them. Let $s_1, \ldots, s_r$ be the new set of (unified) input vertices and $t^1_1, \ldots, t^1_r$ and $t^2_1, \ldots, t^2_r$ be the set of output vertices. Let's call this graph H_r (see Figure 2). An important observation to make here is that H_r is acyclic. This is crucial to our main construction. We call the triple s_i, t^1_i, t^2_i "block" i with *start point* s_i and *end points* t^1_i, t^2_i, $1 \leq i \leq k$. Consider H_r and the $2r$ pairs s_i, t^1_i and s_i, t^2_i (two pairs for each block) as an instance of the EDP problem. We say *block i is fully routed* in a solution to this instance if there are edge-disjoint paths connecting both pairs s_i, t^1_i and s_i, t^2_i in the solution. If only one of these paths exists in the solution then we say *block i is partially routed*. It is easy to see:

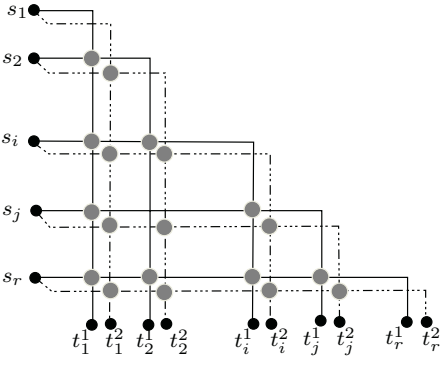


Fig. 2. Construction of H_r from two copies of D_r

Fact 2: Any optimal (integral) solution for EDP on H_k with $2k$ pairs, either contains only one routed block or two partially routed blocks. Furthermore, there is a half-integral solution in which every block is fully routed (with value $\frac{1}{2}$ on each path).

We will use the following technical lemma in our construction.

Lemma 2. *For given positive integers r, k, and g with $r < k$, there is an explicit construction for a k-uniform r-regular hypergraph of girth at least g where the size of the construction (number of vertices) is $O(\frac{k^{2g}}{r})$.*

Proof. We start with an explicit k-regular graph $G(V, E)$ of size at most $O(k^{2g-1})$ and girth at least $2g$. These graphs exist (see, for instance, [17] and the references there). Construct a bipartite graph $G'(A \cup B, E')$ from G where A and B are copies of V with a_i and b_i being the vertices of A and B (respectively) corresponding to vertex $v_i \in V$, and $a_i b_j \in E'$ if and only if $v_i v_j \in E$. It is easy to see that G' is k-regular with girth at least $2g$ and has size $O(k^{2g-1})$. To simplify our calculations, let's assume that k is a power of 2 (this only affects the implicit multiplicative constant in the lemma).

Construct a new bipartite graph $G''(A' \cup B, E'')$ from G' in this way: for each vertex $a_i \in A$ create two vertices a_i^1 and a_i^2 in A'. Then join a_i^1 to half of the neighbors of a_i and join a_i^2 to the other half. Repeating this procedure $\log(k/r)$ times, we get the bipartite graph G'' in which every vertex in A' has degree $\frac{k}{2^{\log(k/r)}} = r$ and every vertex in B has degree k, $|A'| = \frac{k^{2g}}{r}$, $|B| = k^{2g-1}$, and the girth is $2g$. Now we define hypergraph $\mathcal{H}$ with vertex set A' whose edge-set is the set of neighborhoods of vertices of B. This hypergraph is k-uniform, r-regular, with girth g and has size $O(\frac{k^{2g}}{r})$. □

Proof of Theorem 4: Let r, k, and g be some positive integers to be specified later and let $r' = \binom{r}{2}$. Consider a k-uniform r'-regular girth g hypergraph $\mathcal{H}$. Such graphs exist by Lemma 2. The underlying structure of the main graph for the integrality gap is hypergraph $\mathcal{H}$. Let p and q be the number of vertices and hyperedges of $\mathcal{H}$, respectively. Take a set $P_r = \{D_r^1, \ldots, D_r^p\}$ containing p copies of graph D_r (constructed earlier), one corresponding to each vertex in hypergraph $\mathcal{H}$. Also take a set $Q_k = \{R_k^1, \ldots, R_k^q\}$ containing q copies of H_k, one corresponding to each hyperedge of $\mathcal{H}$. For every graph in P_r we fix an arbitrary ordering of its intersection modules (note that the number of intersection modules of D_r is $\binom{r}{2} = r'$; the same as degree of a vertex in $\mathcal{H}$). Similarly, for each graph in Q_k we fix an arbitrary ordering of its blocks (note that the number of blocks of each graph in Q_k is k; the same as the size of a hyperedge in $\mathcal{H}$). Initially, we assign a green flag to every intersection module of every graph in P_r and to every block of every graph in Q_k. Soon we will start modifying the blocks and modules and change their flags to "red". For each pair s_i^j, t_i^j in each copy $D_r^j \in P_r$ add the directed edge $t_i^j s_i^j$ to D_r^j. We call these edges *feedback* edges.

Consider an arbitrary hyperedge $e_i \in \mathcal{H}$ and let $R_k^i \in Q_k$ be the copy of H_k in Q_k which corresponds to hyperedge e_i. Note that R_k^i has k blocks; let's denote these blocks by $b_1^i, \ldots, b_k^i$, where b_λ^i consist of triple $s_\lambda^i, t_\lambda^{1,i}, t_\lambda^{2,i}$. Furthermore,

look at the vertices of hyperedge e_i and find the corresponding copies of D_r in P_r. More precisely, let

$$S_i = \{D_r^{a_j} \in P_r | \text{ the vertex of } \mathcal{H} \text{ corresponding to } D_r^{a_j} \text{ belongs to } e_i, 1 \leq j \leq k\}.$$

Pick the first green block of R_k^i, say $s_\lambda^i, t_\lambda^{i,1}, t_\lambda^{i,2}$ (for some $1 \leq \lambda \leq k$) according to the fixed ordering of the blocks of R_k^i and change its flag to red. Also, pick the first green intersection module of $D_r^{a_\lambda}$ (from its fixed ordering), say $h_{ab}^1, h_{ab}^2, u_{ab}, v_{ab}$ (for some $1 \leq a, b \leq r$) and change its flag to red. Remove vertex h_{ab}^2 and its incident edges (i.e. edges $h_{ab}^1 h_{ab}^2$, $h_{ab}^2 u_{ab}$, and $h_{ab}^2 v_{ab}$) from $D_r^{a_\lambda}$ and add the following edges: $h_{ab}^1 s_\lambda^i$, $t_\lambda^{1,i} u_{ab}$, and $t_\lambda^{2,i} v_{ab}$. We will consider these new three edges (instead of the three edges that were removed from $D_r^{a_\lambda}$) as part of $D_r^{a_\lambda}$. Do this for all the blocks of R_k^i. This process is going to modify (and change the flag from green to red for) one intersection module from each graph in S^i (i.e. $D_r^{a_1}, \ldots, D_r^{a_K}$); one for every block of R_k^i. Repeat the same procedure for all the hyperedges of $\mathcal{H}$ (i.e. for all graphs in Q_k). We obtain a huge directed graph $G_{r,k,g}$, which has constant degree and $O(r^2 p + k^2 q)$ vertices. Note that, since each graph $R_k^i \in Q_k$ is acyclic, every cycle in $G_{r,k,g}$ must contain one of the feedback edges.

The basic idea behind the construction is that intersection modules in copies of D_r (graphs in P_r) are now replaced with "blocks" of copies of H_k (graphs in Q_k) and in order to go from h_{ab}^1 to u_{ab} in the intersection module of $D_r^{a_\lambda}$, we have to go from s_λ^i to $t_\lambda^{1,i}$ in a "block" of R_k^i. For the moment, assume that:

(i) all the blocks where completely independent of each other i.e. they were not part of the same graph and therefore there was no way to start at the start point of a block b_i (of a copy of H_k) and end at an end point of another block b_j.
(ii) among all the blocks, only an ϵ-fraction could be (partially or fully) routed, and for the other $(1-\epsilon)$-fraction no routing existed at all.

This would imply that $G_{r,k,g}$ has no more than ϵrp cycles. The reason is that each cycle must contain a feedback edge, and so goes from some s_α in a copy of D_r in P_r to t_α in the same copy. Therefore, it goes through at least r blocks (previously intersection modules), since each cycle in a copy of D_r uses at least r intersection modules. This would give us the required gap. Fortunately, the assumption (i) above is easy to prove (by Fact 2), i.e. a large fraction of all of the blocks of graphs in Q_k do not have any routing (neither partial nor full). But the trouble is that the second assumption is not correct. That is, the blocks are not completely independent as we assumed, and they appear in groups of size k in one graph (a copy of H_k in Q_k). For this reason, cycles in $G_{r,k,g}$ may have complicated structures and go through several copies D_r^i's in P_r. For instance, a cycle C may start (as a path) at some vertex s_α in a copy D_r^i in P_r (D_r^i corresponds to vertex $v_i \in \mathcal{H}$) and then at some vertex $h_{ab}^1 \in D_r^i$ the path enters the start point of a block b_x in a graph R_k^j (which is a copy of H_k in Q_k). But instead of going out from an end point of the same block b_x (of R_k^j) it goes (within R_k^j) to an end point of another block b_y of R_k^j. We call this situation

a *jump between blocks* of R_k^j. This way, the path may end-up in another copy $D_r^{i'}$ (before going back to s_α). Looking from a higher level at the underlying hypergraph structure (which has a structure like $\mathcal{H}$), we can think of this path as going from vertex v_i (graph $D_r^i \in P_r$) to $v_{i'}$ (graph $D_r^{i'} \in P_r$) in $\mathcal{H}$ through hyperedge e_j (through graph $R_k^j \in Q_k$ by starting at the start point of one block and going down to an end point of another block of R_k^j). But if this happens, since the start point (s_α) is in D_r^i, this path must eventually come back to t_α in D_r^i (because $t_\alpha \in D_r^i$ is the only vertex that has an edge to s_α). However, because $\mathcal{H}$ has girth g, the path has to go through at least g other graphs in P_r before getting back into D_r^i. Therefore, the cycle contains at least $\Omega(g)$ edges from the graphs in P_r. We call these cycles (that go through several graphs in P_r) *long* cycles (because g is going to be large) and those that are within one graph of P_r (and so do not jump between blocks of graphs in Q_k), *short* cycles. This implies that the total number of long cycles can be at most a fraction $\frac{1}{g}$ of the total number of edges in the graphs in P_r. If g is large and r is small the total number of short and long cycles will be small.

Lemma 3. *$\nu_e^*(G_{r,k,g}) \in \Omega(rp)$, that is, $G_{r,k,g}$ has a fractional cycle packing solution of value $\Omega(rp)$.*

Proof. Recall that by Fact 1, there is a half-integral solution (for EDP problem) in any instance D_r, which contains one half-integral path for each pair s_i, t_i. If we add edges $t_i s_i$ (for $1 \leq i \leq r$) to D_r then there are at least r half-integral cycles in D_r. In this fractional solution, we route exactly two (half-integral) cycles through each intersection module.

We do have the feedback edges in $G_{r,k,g}$ (in every graph $D_r^j \in P_r$). Also, by Fact 2, for every graph $R_k^i \in Q_k$ there is a half-integral fractional solution in which all the blocks in R_k^i are fully routed (with value $\frac{1}{2}$). Therefore, all the blocks in all graphs in Q_k (which have replaced all the intersection modules in graphs in P_r) are fully routed (with value $\frac{1}{2}$). These two imply that each (modified) graph $D_r^j \in P_r$ has r half-integral (short) cycles (where parts of the fractional cycles go through blocks of the graphs in Q_k). Since there are p graphs in P_r we get $\Omega(rp)$ half-integral cycles. □

Lemma 4. *$\nu_e(G_{r,k,g}) \in O(\frac{q}{r} + \frac{r^2 p}{g})$.*

Proof sketch. By Fact 2, for every graph $R_k^i \in Q_k$, there is at most two blocks that can be (partially or fully) routed. So over all graphs in Q_k, there are at most $2q$ blocks that can be (partially or fully) routed. Since blocks have replaced the intersection modules of the graphs in P_r and every short cycle in a graph in P_r goes through at least r blocks, plus the fact that at most two cycles can go through any routed block, there can be at most $4q/r$ directed short cycles in the graphs of P_r in $G_{r,k,g}$.

Now we upper bound the number of long cycles. Because every graph in P_r has constant degree and $O(r^2)$ vertices, the total number of edges of $G_{r,k,g}$ that are parts of the graphs in P_r is $O(r^2 p)$. Therefore, by the arguments before

Lemma 3, there are at most $O(\frac{r^2p}{g})$ long cycles in $G_{r,k,g}$. Thus the total number of short and long cycles is $O(\frac{q}{r} + \frac{r^2p}{g})$. □

Recall that for hypergraph $\mathcal{H}$, the number of vertices p and hyperedges q are in $O(\frac{k^{2g}}{r})$ and $O(k^{2g-1})$, respectively. Let r be some (not too small) constant and $k = g$. This implies that $p \in O(k^{2k})$ and $q \in O(k^{2k-1})$. The total number of vertices n in graph $G_{r,k,g}$ is $O(r^2p + k^2q)$ which is $O(k^{2k+1})$. By Lemmas 3 and 4, the integrality gap is at least $\Omega((rp)/(\frac{q}{r} + \frac{r^2p}{g}))$ which is $\Omega(k)$. This is $\Omega(\frac{\log n}{\log\log n})$, which completes the proof of Theorem 4. □

Combining Theorems 3 and 4 and noting that the constructions in Theorem 3 have size polynomial, we obtain:

Corollary 1. *Directed VDC has an integrality gap of* $\Omega(\frac{\log n}{\log\log n})$.

The construction for the hardness result has similar structure and uses the hardness of directed EDP by Ma and Wang [19] which is based on PCP theorem [1, 2] together with Raz [20] parallel repetition theorem.

Theorem 5. *[19] For any* $\epsilon > 0$*, directed EDP cannot be approximated within ratio* $2^{-\log^{1-\epsilon} n}$ *unless* $NP \subseteq DTIME(2^{polylog(n)})$.

A careful analysis of proof of Theorem 5 reveals that in fact their proof implies the following stronger version:

Theorem 6. *Given an instance* I *of directed EDP, which consists of an acyclic digraph* G *(on* n *vertices) and* k *source-sink pairs* $(s_1, t_1), \ldots, (s_k, t_k)$ *in* G*, where* $k \in \Omega(n^\delta)$ *for some absolute* $\delta > 0$*, it is quasi-NP-hard to decide between the following two cases:*

1. *(Yes-instance) All pairs* (s_i, t_i) *can be routed by disjoint paths, or*
2. *(No-instance) At most a fraction* $2^{-\log^{1-\epsilon} n}$ *of the pairs can be routed.*

Theorem 7. *For any* $\epsilon > 0$*, there is no* $O(\log^{1-\epsilon} n)$*-approximation for EDC unless* $NP \subseteq DTIME(2^{polylog(n)})$.

Proof. Let I_{EDP} be an instance of (directed) EDP as in Theorem 6 which consists of a directed acyclic graph G and k pairs (s_i, t_i), $1 \le i \le k$. Take two copies of I_{EDP}, named I_P^1 and I_P^2 and identify the source $s_i^1 \in I_P^1$ with $s_i^2 \in I_P^2$ and call this new vertex s_i $(1 \le i \le k)$. Denote this new graph by H_k, with $2k$ source-sink pairs s_i, t_i^1 and s_i, t_i^2, $1 \le i \le k$. As in the construction of the integrality gap, we name the triple s_i, t_i^1, t_i^2 *block* i of H_k with start point s_i and end points t_i^1 and t_i^2. In a solution to EDP problem with instance H_k and the $2k$ pairs s_i, t_i^1 and s_i, t_i^2 $(1 \le i \le k)$ we say block i is fully routed if there are two paths, one from s_i to t_i^1 and one from s_i to t_i^2, in the solution. If only one of these paths exists then we say block i is partially routed. If none of them exists block i is not routed at all. Again, the fact that H_k is acyclic will be crucial in the analysis of our

construction. Let r and g be some positive integers (to be specified later) and take a $\binom{r}{2}$-regular k-uniform hypergraph $\mathcal{H}$ with girth g. As before, let p and q be the number of vertices and hyperedges of $\mathcal{H}$, respectively. We construct graph $G_{r,k,g}$ whose underlying structure is hypergraph $\mathcal{H}$ in the same manner we did in Theorem 4 except that now we use copies of H_k (defined above) to place in Q_k. The rest of the construction remains the same. That is, we take p copies of D_r and put them in the set P_r and then replace the intersection modules of them with blocks of copies of H_k in Q_k in the same manner. Let $P_r = \{D_r^1, \ldots, D_r^p\}$ and $Q_k = \{R_k^1, \ldots, R_k^q\}$. We define *short* and *long* cycles in $G_{r,k,g}$ in the same way as we did in Theorem 4.

If I_{EDP} is a *Yes* instance then all the k blocks in H_k can be fully routed. This means that every block of every $R_k^i \in Q_k$ can be fully routed. So for every graph $D_r^j \in P_r$ there are r disjoint paths from s_i to t_i, one for each $1 \leq i \leq r$, and because of the existence of feedback edges (connecting t_i to s_i) we have r edge-disjoint cycles in each $D_r^j \in P_r$. This gives a total of $\Omega(rp)$ edge-disjoint cycles.

If I_{EDP} is a *No* instance then at most a fraction $2^{-\log^{1-\epsilon} n}$ of the k pairs can be routed. Since $k \in \Omega(|I_{EDP}|^{\delta})$, this fraction is at most $2^{-\left(\frac{\log k}{\delta}\right)^{1-\epsilon}}$, which we name it α. So, at most $2\alpha k$ blocks in each graph in Q_k, and therefore, at most $2\alpha kq$ blocks over all the graphs in Q_k can be (fully or partially) routed. Because each short cycle in a graph $D_r^i \in P_r$ goes through r blocks (previously intersection modules), the number of short cycles over all graphs in P_r is at most $2\alpha kq/r$. The same argument we had in Theorem 4 for long cycles implies that the number of long cycles here is at most a $\frac{1}{g}$ fraction of the total number of edges in all graphs in P_r. This is at most $O(\frac{r^2p}{g})$. All together, the number of short and long cycles is $O(\frac{2\alpha kq}{r} + \frac{r^2p}{g})$, which is $O(rp(2\alpha + \frac{r}{g}))$, because $q = \frac{r^2p}{k}$.

The above arguments, together with Theorem 6 imply that deciding between $\Omega(rp)$ cycles and $O(rp(2\alpha+\frac{r}{g}))$ cycles in $G_{r,k,g}$ is quasi-NP-hard. Equivalently, it is quasi-NP-hard to have an approximation algorithm with factor $\Omega(1/F(r,k,g))$ where $F(r,k,g) = 2\alpha + \frac{r}{g}$. Let r be a (not too small) constant and $g = O(\log^c k)$ for an arbitrary large constant $c > 0$. This implies that the hardness factor (i.e. $1/F(r,k,g)$) is $\Omega(\log^c k)$. With this setting of parameters, hypergraph $\mathcal{H}$ has $p = O(k^{\log^c k})$ vertices and $q = \frac{rp}{k} = O(k^{\log^c k - 1})$ edges. So, if N denotes the number of vertices of $G_{r,k,g}$ (i.e. the size of construction), then it will be at most $O(k^{\log^c k})$ (for the vertices in graphs in P_r) plus $O(k^{\frac{2}{\delta}+\log^c k})$ (for the vertices in graphs in Q_k). So overall, $N \in O(k^{\frac{2}{\delta}+\log^c k})$, which is quasi-polynomial in the size of input (instance I_{EDP}). Rewriting the hardness factor $\Omega(\log^c k)$ in terms of N gives a hardness of $\Omega(\log^{\frac{c}{c+1}} N)$. □

Under a stronger complexity assumption that for some sufficiently small $\delta > 0$, $NP \not\subseteq DTIME(2^{n^{\delta}})$, we can improve the hardness result to $\Omega(\frac{\log n}{\log \log n})$.

References

1. S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, *Proof verification and hardness of approximation problems*, J. of the ACM, 45(3):501–555, 1998.
2. S. Arora and S. Safra, *Probabilistic checking of proofs: A new characterization of NP*, J. of the ACM, 45(1):70-122, 198.
3. V. Bafna, P. Berman, and T. Fujito, *Constant ratio approximation of the weighted feedback vertex set problem for undirected graphs*, In Proc. of ISAAC'95, Lecture Notes in Computer Science No. 1004, pages 142-151, Springer-Verlag, 1995.
4. P. Balister, *Packing digraphs with directed closed trials*, Combinatorics, Probability, and Computing, 12(1):1–15, 2003.
5. A. Becker and D. Geiger, *Approximation algorithms for the loop cutset problem*, In Proc. of 10th Conf. on Uncertainty in ARtificial Intelligence, pages 60-68, 1994.
6. A. Carpara, A. Panconesi, and R. Rizzi, *Packing cycles in undirected graphs*, J. Algorithms 48(1): 239–256, 2003.
7. B. Carr and S. Vempala, *Randomized meta-rounding*, In Proc. of STOC 2000.
8. C. Chekuri and S. Khanna, *Edge Disjoint Paths Revisited*, Proc. of SODA 2003.
9. J. Cheriyan and M.R. Salavatipour, *Hardness and Approximation Results for Packing Steiner Trees Problems*, In Proc. of ESA'04.
10. F. A. Chudak, M. X. Goemans, D. S Hochbaum, and D. P. Williamson, *A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs*, Operations Research Letters, 22, 111–118, 1998.
11. P. Erdös and L. Pósa, *On the maximal number of disjoint circuits of a graph*, Publ. Math. Debrecen 9:3–12, 1962.
12. G. Even, J. Naor, B. Schieber, and L. Zosin, *Approximating minimum subset feedback sets in undirected graphs with applications*, SIAM J. Discrete Math. 13(2):255-267, 2000.
13. G. Even, J. Naor, B. Schieber, and M. Sudan, *Approximating minimum feedback sets and multicuts in directed graphs*, Algorithmica 20(2): 151–174, 1998.
14. V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis, *Near-Optimal Hardness Results and Approximation Algorithms for Edge-Disjoint Paths and Related Problems*, J. of Computer and System Sciences 67(3):473–496, 2003. Earlier version in STOC'99.
15. K.Jain, M.Mahdian, M.R.Salavatipour, *Packing Steiner trees*, In Proc. SODA 2003.
16. M. Krivelevich, Z. Nutov, R. Yuster, *Approximation algorithms for cycle packing problems*, To appear in Proc. of SODA'05.
17. F. Lazebnik, V.A. Ustimenko and A.J. Woldar, *New upper bounds on the order of cages*, Electronic Journal of Combinatorics, Volume 14, R13: 1–11, 1997.
18. L.C.Lau, *An approximate max-Steiner-tree-packing min-Steiner-cut theorem*, In Proc. 45th IEEE FOCS 2004.
19. B. Ma and L. Wang, *On the inapproximability of disjoint paths and minimum steiner forest with bandwidth constraints*, J. of Computer and Systems Sciences, 60(1):1–12, 2000.
20. R. Raz, *A parallel repetition theorem*, SIAM J. of Computing, 27(3): 763–803, 1998.
21. P. D. Seymour, *Packing directed circuits fractionally*, Combinatorica 15(2):281–288, 1995.

A Combinatorial Algorithm to Find a Maximum Even Factor*

Gyula Pap**

Department of Operations Research, Eötvös University,
Budapest, Hungary
gyuszko@cs.elte.hu

Abstract. We give a new, algorithmic proof for the maximum even factor formula which can be converted into a polynomial time combinatorial algorithm to solve the maximum even factor problem. In several aspects, the approach is similar to Edmonds' Matching Algorithm, but there is a significant difference.

1 Introduction

W.T. Tutte [15] characterized the existence of a perfect matching in a graph, then C. Berge gave a min-max formula in [2] to determine the maximum size of a matching – the so-called Berge-Tutte formula. In [6] J. Edmonds gave an efficient algorithm to determine a maximum matching, which also implies a structural description – independently found by T. Gallai [7] – called the Gallai-Edmonds decomposition.

W.H. Cunningham and J.F. Geelen [4] investigated separation algorithms for the matchable set polytope, first studied by E. Balas and W. Pulleyblank in [1]. In [4] they concluded that a strongly polynomial time separation algorithm can be given, provided there is a polynomial time algorithm for solving a combinatorial problem which they called the optimal path-matching problem. They settled this by giving polynomial algorithms for the optimal path-matching problem, which is in fact a direct generalization of the matching problem. Tutte's original approach related matchings to the rank of a matrix over indeterminates – now known as the Tutte-matrix – Cunningham and Geelen noticed that the maximal value of a path-matching is equal to the rank of an analogue matrix. In general, it is an unsolved problem to determine the rank of a matrix A in polynomial time, if the entries of A are linear expressions of algebraically independent indeterminates. However, evaluations of the indeterminates with random integers yield a polynomial-time randomized algorithm, using the fact that the rank of a matrix

* Research supported by the Hungarian National Foundation for Scientific Research Grant, OTKA T037547.

** The author is a member of the MTA-ELTE Egerváry Research Group (EGRES). Supported by European MCRTN Adonet, Contract Grant No. 504438.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 66–80, 2005.

over the integers can be determined in polynomial-time; see [9]. This provides a polynomial time randomized algorithm for maximum matching, optimal path-matching; Cunningham and Geelen gave two different deterministic algorithms. In [4] they showed a polyhedral description which admits polynomial time separation, then the optimal path-matching problem can be solved via the ellipsion method. Later in [5] they gave an algorithm based on deterministic evaluations of the Tutte-matrix. Both algorithms return a maximum path-matching along with a so-called minimal "stable pair", this gives a min-max formula for optimal path-matchings.

As a direct extension of the Berge-Tutte formula, A. Frank and L. Szegő [8] gave a simplified min-max formula for optimal path-matchings. A Gallai-Edmonds-type structural description was given by B. Spille and L. Szegő [13]. Both results are formulated so that known results about matchings are direct special cases, in fact techinques useful for matchings are used in their proofs. So it was a challange to develop an algorithm that follows lines similar to those in Edmonds' matching algorithm. B. Spille and R. Weismantel [14] constructed such an algorithm to find an optimal path-matching.

In [3] Cunningham introduced the so-called maximum even factor problem in digraphs, which is yet another generalization of maximum matching and optimal path-matching problems. The maximum even factor problem is NP-hard, but the problem is tractable when restricted to the class of odd-cycle-symmetric digraphs – this includes the path-matching problem as a special case. We refer to the maximum even factor problem as the restricted problem to odd-cycle-symmetric graphs. This problem may be represented by the rank of a Tutte-type matrix, the algorithm in [5] can be used to compute this rank.

The simplified formula of Frank and Szegő on path-matchings was extended to even factors by G. Pap and L. Szegő [12]. In fact, in [12] the formula is only formulated in a tighter class of graphs called "weakly symmetric", but the method also works for odd-cycle-symmetric graphs, this was already known by the authors at the time of publication. The aim of this paper is to give a new proof of the min-max formula presented in [12] for odd-cycle-symmetric graphs, this proof can be converted into a polynomial-time algorithm for the maximum even factor problem. Our approach is different from any approaches mentioned above. It bears some characteristics of Edmonds' matching algorithm, but there are significant differences, too. When we apply this algorithm to solve a maximum matching problem, we do not get back to Edmonds' algorithm. The concept of alternating forests is replaced by a weaker structure which is easier to maintain.

The paper has the following structure: after the notation is introduced, we give an algorithmic proof for the Berge-Tutte type Theorem 1; next, this is extended to a Gallai-Edmonds type structural description. In the end we make some concluding remarks on the concept of the paper, relating it to well-known concepts for matchings, accompanied by some examples.

2 Basic Definitions for Even Factors

In this section we summerize notions and notation used in the paper. Consider a digraph $D = (V, A)$ without loops, a *cycle (path)* is the arc-set of a closed (unclosed) directed walk without repetition of arcs or nodes. *A path-cycle-factor* is the arc-set $M \subseteq A$ of a subgraph of G which is a node disjoint union of paths and cycles. Equivalently, a path-cycle-factor is a set of edges with in- and out-degree at most 1 in each node. We call an arc $e = uv \in A$ *symmetric* if $vu \in A$, otherwise e is *asymmetric*. A cycle or a path is *even (odd)* if it consists of an even (odd) number of arcs; a cycle is *asymmetric*, whenever it has at least one asymmetric arc. An *even factor* is a path-cycle-factor which has no odd cycle, let $\nu(D)$ denote the maximum cardinality of an even factor. An even factor is *perfect* if each node is covered with an even cycle.

A min-max formula was given in [12] by Pap and Szegő for $\nu(D)$ in a special class of digraphs D called weakly symmetric. A graph is called *weakly symmetric* (a definition introduced in [3]) if all cycles in the graph are symmetric – or equivalently, if all edges inside a strong component are are symmetric. The results and the method in [12] also work in a slightly larger class of graphs called odd-cycle-symmetric: D is called *odd-cycle-symmetric* if there is no asymmetric odd cycle – or equivalently, if each odd cycle in D is symmetric.

Throughout the paper we only consider odd-cycle-symmetric graphs, we show that determining $\nu(D)$ for odd-cycle-symmetric graphs is polynomially solvable. The restriction is in fact necessary, since for an arbitrary graph D it is NP-complete to determine $\nu(D)$ [3]. In some papers, we have used the terms "hardly symmetric graphs" for the notion called odd-cycle-symmetric graphs in this paper.

Some further definitions, notation: for a set $X \subseteq V$ let $\Gamma^+_D(X) := \{x \in V - X : \exists y \in X, yx \in A\}$ and $\varrho_D(X) = |\{uv \in A : u \in V - X, v \in X\}|$ and $\delta_D(X) = |\{uv \in A : u \in X, v \in V - X\}|$. For a path-cycle-factor M let $V^+(M) := \{v \in V : \delta_M(v) = 1\}$ and $V^-(M) := \{z \in V : \varrho_M(v) = 1\}$, so the set $V - V^+(M)$ consists of the *M-source-nodes* and $V - V^-(M)$ consists of the *M-sink-nodes*. For an arc set $F \subseteq A$ then let $\overleftrightarrow{F} := F \cup \{uv : vu \in F\}$, hence if $C \subseteq A$ is an odd cycle then $\overleftrightarrow{C} \subseteq A$. A set $U \subseteq V$ *induces* the subgraph $D[U] = (U, A[U])$ where $A[U] = \{uv \in A : u, v \in U\}$. $D - U := D[V - U]$. For a set $U \subseteq V$ we denote the *contracted graph* by D/U having node set $V/U = V - U + \{U\}$ and arc-set A/U given by deleting the arcs in $A[U]$ and identifying the nodes in U by $\{U\}$ (we will use contractions only if $D[U]$ is connected, thus this definition is equivalent with the usual contraction of the arcs in $A[U]$). For $uv \in A$ we call v the *head* of arc uv, while u is called the *tail* of arc uv; uv *leaves* u and *enters* v.

An undirected graph $G = (V, E)$ is called *factor-critical* if for all $v \in V$ the graph $G - v$ has a perfect matching. (For a survey on matching theory, see L. Lovász and M.D. Plummer [10].) A digraph is called *symmetric-critical* if all its arcs are symmetric and the underlying undirected graph is factor-critical. In a digraph D a set $S \subseteq V$ is called a *source-component* if $D[S]$ is strongly connected and $\varrho_D(S) = 0$. Let $\sigma(D[X])$ denote the number of those source-

components in $D[X]$ which are symmetric-critical. For example a source-node is a symmetric-critical source-component.

(!) Recall that in the well-known Berge-Tutte formula we use the number of odd components of $G-X$, here the definition of $\sigma(D[X])$ is concerned with $G[X]$. Notice, if we orient the edges of a factor-critical graph, we do not necessarily get a symmetric-critical digraph: it will only be symmetric-critical if all edges are symmetric. **(!)**

Definition 1. *The* deficiency *of an even factor M is* $\mathrm{def}_D M := |V| - |M|$, *which is non-negative, of course. The* deficiency *of a set $X \subseteq V$ is defined by* $\mathrm{def}_D X := \sigma(D[X]) - |\Gamma_D^+(X)|$. *Let us use the notation* $\tau_D(X) := |V| + |\Gamma_D^+(X)| - \sigma(D[X]) = |V| - \mathrm{def}_D X$ *for a node set $X \subseteq V$.*

To prove the main theorem 1, in the next section we will use the following well-known statements about factor-critical undirected graphs:

Lemma 1. *If $G = (V, E)$ is an undirected graph, for some $U \subseteq V$ the induced subgraph $G[U]$ is factor-critical and G/U is factor-critical, then G is factor-critical, too.*

Lemma 2. *If $G = (V, E)$ is a factor-critical undirected graph and $s, t \in V$ then*

a) if $s \neq t$ then there is an even path P from s to t and a perfect matching M in $G - V(P)$,

b) if $s = t$ then there is a perfect matching M in $G - s$ (a degenerate version of a)).

Proof. Let M_s, M_t be perfect matchings in $G - s$, $G - t$, respectively. Then $M_s \cup M_t$ consists of alternating cycles – which have even length – and one alternating s, t path P on an even number of edges – in case of $s = t$ P has no edges. In the alternating cycles let us choose the edges of M_s to form the perfect matching M on $V - V(P)$.

3 The Main Theorem with a Constructive Proof

In this section, we give a simple, algorithmic proof of the following theorem.

Theorem 1 (Pap, Szegő [12]). *If $D = (V, A)$ is an odd-cycle-symmetric digraph, then*

$$\nu(D) = \min_{X \subseteq V} |V| + |\Gamma_D^+(X)| - \sigma(D[X]). \tag{1}$$

The easy part of the proof is to see that the left hand side in (1) is at most the right hand side. To this end we show that for any even factor M and any set $X \subseteq V$ we have $|M| \leq \tau_D(X)$. Let

$$M_1 := \{vz \in M : v, z \in X\} = M[X], \tag{2}$$

$$M_2 := \{vz \in M : v \in X, z \in V - X\}, \tag{3}$$

$$M_3 := \{vz \in M : v \in V - X\}. \tag{4}$$

It is easy to see that M_1, M_2, M_3 is a partition of M.

Lemma 3. *The following inequalities hold:*

$$|M_1| \leq |X| - \sigma(D[X]), \tag{5}$$
$$|M_2| \leq |\Gamma_D^+(X)|, \tag{6}$$
$$|M_3| \leq |V| - |X|. \tag{7}$$

Proof. First we show that if $S \subseteq X$ is a symmetric-critical component in $D[X]$ then there is a node s in S for which $\varrho_{M_1}(s) = 0$. Suppose for contradiction, for each node v in S there is an arc in M_1 with head v, then S would be covered by cycles in M_1 – and since a symmetric-critical graph has odd number of nodes – there would be an odd cycle in M. Thus there must be at least $\sigma(D[X])$ nodes s in X with $\varrho_{M_1}(s) = 0$, this proves inequality (5). Each arc in M_2 has head in $\Gamma_D^+(X)$, this proves inequality (6). Each arc in M_3 has tail in $V - X$, this proves inequality (7).

The sum of inequalities (5)–(7) gives $|M| \leq \tau_D(X)$.

Definition 2. *A set $X \subseteq V$ is a* verifying set for *an even factor M if $|M| = \tau_D(X)$, or equivalently* $\mathrm{def}_D M = \mathrm{def}_D X$.

Lemma 4. *If M is an even factor with a verifying set X – i.e. equality holds in (1) – then each of (5)-(7) holds with equality. In particular, for each node $v \in V - X$ we have $\delta_M(v) = 1$.*

Proof. Equality in (7) implies $\delta_M(v) = 1$ for the nodes v in $V - X$.

As a preparation for the more difficult part of the proof we need to study some properties of symmetric-critical subgraphs and even factors.

Lemma 5. *If $D = (V, A)$ is symmetric-critical and $s, t \in V$ are two not necessarily distinct nodes, then there is an even factor M_{st} for which*

1. *$|M_{st}| = |V| - 1$,*
2. *M_{st} has no arc entering s and no arc leaving t,*
3. *M_{st} consists of even cycles and an even s,t path P_{st} (in case of $s = t$ this path has length zero).*

Proof. Let us apply Lemma 2 for the underlying undirected graph. M_{st} is constructed by taking the edges in M in both directions and taking edges on the path P in direction s to t.

Lemma 6. *In an odd-cycle-symmetric digraph $D = (V, A)$, if $C \subseteq A$ is an odd cycle then $D[V(C)]$ is symmetric-critical.*

Proof. By definition, the arcs of C must be symmetric, thus $\overleftrightarrow{C} \subseteq A$. For any arc $ab \in A[V(C)]$ there is an even path $P \subseteq \overleftrightarrow{C}$ for which $P + ab$ is an odd cycle. Thus ab must be symmetric.

Definition 3. *For a node-set $U \subseteq V$, a directed path $P \subseteq A$ is called an* odd ear on U *if the first and last nodes of P are in U, the inner nodes of P are in $V - U$, and $|P|$ is odd.*

Lemma 7. *If $D = (V, A)$ is odd-cycle-symmetric and $D[U]$ is symmetric-critical for some $U \subseteq V$, then the contracted graph D/U is odd-cycle-symmetric.*

Proof. Odd cycles in D/U not incident with $\{U\}$ are symmetric, since $D/U - \{U\}$ is isomorfic to $D - U$. Consider an odd cycle C of D/U incident with $\{U\}$. The set $C' \subseteq A$ of arcs corresponding to the arcs of C must be an odd ear on U – say C' has first node t and last node s. By Lemma 5 there must be an even path P_{st} from s to t in $D[U]$, then $C' \cup P_{st}$ is an odd cycle in D. So all arcs in C' must be symmetric, then all arcs in C must be symmetric in D/U, too.

Lemma 8. *Consider an odd-cycle-symmetric digraph D, suppose node-set $U \subseteq V$ induces a symmetric-critical subgraph $D[U]$, Y induces a symmetric-critical subgraph in D/U with $\{U\} \in Y$. Then the pre-image of Y, i.e. $Z = Y - \{U\} \cup U$ induces a symmetric-critical subgraph in D.*

Proof. By Lemma 1 we only need to show that each arc in $D[Z]$ is symmetric. Since all arcs in $D[U]$ and $D/U[Y]$ are symmetric, only arcs $ab \in A[Z]$ with $|\{a, b\} \cap U| = 1$ need to be checked. We show that ab is contained in an odd cycle of D. By Lemma 5 there is an even path P_{ba} in $D/U[Y]$, then $C = P_{ba} + ab$ is an odd cycle in D/U. The set of arcs $C' \subseteq A$ corresponding to C give an odd ear on U, suppose this odd ear has first node t and last node s. By Lemma 5 there is an even path P_{st} in $D[U]$, then $C' + P_{st}$ is an odd cycle in D.

Definition 4. *We say an even factor N* fits the odd cycle C *if $|N \cap C| = |V(C)| - 1$ and $\delta_N(V(C)) = 0$.*

Observe, if N is an even factor, C is an odd cycle, and N_C is the set of arcs of N adjacent with a node in $V(C)$, then N fits C if and only if either $N_C = C - ab$ (for an arc $ab \in C$) or $N_C = C - ab + cb$ (for an arc $ab \in C$ and an arc $cb \in A$ with $c \notin V(C)$). Consider an even factor N which fits the odd cycle C, let us summerize the consequences of the above Lemmas on N and C: $V(C)$ induces a symmetric-critical subgraph, $D/V(C)$ is hardly symmetric, and N/C is an even factor in $D/V(C)$.

The following theorem describes the key property of the contraction of an odd cycle for even factors, which is the analogue of a known property of matchings, see Section 5.

Theorem 2. *Suppose $D = (V, A)$ is odd-cycle-symmetric, N is an even factor in D which fits the odd cycle C; define $D' = D/V(C)$ and $N' = N/V(C)$.*

1. *If N is maximum, then N' is maximum.*
2. *If X' is a verifying set for N', then the pre-image X is a verifying set for N.*

It is easy to see that the first assertion in this theorem follows from Lemma 9, while the second follows from Lemma 10. For technical reasons in the algorithm, these claims are formulated a little bit stronger than needed for Theorem 2.

Lemma 9. *Suppose $D = (V, A)$ is odd-cycle-symmetric, C is an odd cycle in D, and M is an even factor in $D/V(C)$. Then*

1. *there is an even factor of size $|M| + |V(C)| - 1$ in D,*
2. *$\nu(D) \geq \nu(D/V(C)) + |V(C)| - 1$.*

Proof. The second statement follows from the first, we need to prove the first. Let M' denote the set of arcs in D corresponding to arcs in M. Then M' contains at most one arc entering U, let this arc be $s's$ if any, otherwise choose $s \in V(C)$ arbitrarily. Similarly, M' contains at most one arc leaving $V(C)$, let this arc be tt' if any, otherwise choose $t \in V(C)$ arbitrarily. Let M_{st} be the even factor in $D[V(C)]$ as in Lemma 5, we show that $M' \cup M_{st}$ is an even factor of cardinality $|M| + |V(C)| - 1$.

The cardinality is as indicated, since M' and M_{st} are disjoint, it is also easy to see that $M' \cup M_{st}$ is a path-cycle-factor. The only way we could get an odd cycle would be using the arcs $s's$ and tt'. Let C be the cycle in $M' \cup M_{st}$ using these edges, then C/P_{st} is a cycle in M and P_{st} is even, thus C must be even, too.

Lemma 10. *Suppose N is an even factor in D which is nice with some odd cycle C; define $D' = D/V(C)$ and $N' = N/V(C)$. As already observed, N' is an even factor in D'. If X' is a verifying set in D' for N', then $\{C\} \in X'$ and the pre-image $X := X' - \{C\} \cup V(C)$ is a verifying set for N.*

Proof. Since X' is a verifying set for N', N' is a maximum even factor. By definition $\delta_{N'}(\{C\}) = 0$, thus by Lemma 4 we get that $\{C\} \in X'$. Consider a symmetric-critical source-component Q' in $D'[X']$, if $\{C\} \notin Q'$ then Q' is a symmetric-critical source-component in $D[X]$, too. If $\{C\} \in Q'$ then we claim that $Q := Q' - \{C\} \cup V(C)$ induces a symmetric-critical source-component in $D[X]$, which can be seen as follows: Since $D'[Q']$ is a source component in $D'[X']$, $D[Q]$ must be a source component in $D[X]$; moreover $D'[Q']$ is symmetric-critical, $D[V(C)]$ is symmetric-critical, then by Lemma 8 $D[Q]$ is symmetric-critical.

Thus $D[X]$ has at least as many symmetric-critical source-components as $D'[X']$, that is $\sigma(D[X]) \geq \sigma(D'[X'])$. Furthermore $\{C\} \in X'$ implies $\Gamma^+_D(X) = \Gamma^+_{D'}(X')$ and $|X| = |X'| - |V(C)| + 1$.

$$\begin{aligned} |N| &= |N'| + |V(C)| - 1 = \nu(D') + |V(C)| - 1 = \\ &= |V/C| + |\Gamma^+_{D'}(X')| - \sigma(D'[X']) + |V(C)| - 1 \geq \\ &\geq |V| + |\Gamma^+_D(X)| - \sigma(D[X]) \geq |N|. \end{aligned}$$

The following definitions present the tool to find N fitting an odd cycle C.

Definition 5. *Let M be a fixed even factor in $D = (V, A)$, recall the definition $V^+(M) := \{v \in V : \delta_M(v) = 1\}$, let $K^+ := V - V^+(M)$ – i.e. K^+ is the set of M-sink-nodes. A sequence $W = (v_0, e_0, v_1, e_1, \ldots, v_{n-1}, e_{n-1}, v_n)$ is called an* M-alternating walk *if $v_0 \in K^+$, $e_i = v_i v_{i+1} \in A$ if i is even, $e_i = v_{i+1} v_i \in M$ if i is odd. Here n is called the* length *of W, v_0 is the* first node *of W and v_n is the* last node *of W. W is called* even/odd *by the parity of its length. Let $A(W) = \{e_i : 0 \leq i \leq n-1\}$ denote the* set of arcs in *W. A node v_i with i even/odd is called an* even/odd node *of W.*

Definition 6. *An M-alternating walk $W = (v_0, e_0 v_1, e_1, \ldots, v_{n-1}, e_{n-1}, v_n)$ is called* special *if its even nodes are pairwise distinct, and its odd nodes are pairwise distinct. The* starting segment *of a walk $(v_0, e_0, v_1, e_1, \ldots, v_n)$ of length k is $(v_0, e_0, v_1, e_1, \ldots, v_k)$. Notice, the starting segment of a special M-alternating walk is a special M-alternating walk.*

Lemma 11. *If for some nodes $w, z \in V$ there is an even M-alternating walk from w to z, then there is a special even M-alternating walk from w to z, too.*

Proof. If an M-alternating walk W is not special, then $v_i = v_j$ (for some $i < j$, $i \equiv j \mod 2$), so a shorter M-alternating walk can be constructed by deleting the section from v_i to v_j. So it is easy to see, that a shortest M-alternating walk of even length from v_0 to v_n must be special.

Definition 7. *Let $L = L(D, M)$ be the set of nodes in V which arise as a last node of some even alternating walk.*

Given an even factor M in a digraph D, the set $L(D, M)$ can be determined quite easily – by using an ordinary breadth first search in an auxiliary bipartite graph.

Lemma 12. *For a special even M-alternating walk W, $M \Delta A(W)$ is a path-cycle-factor.*

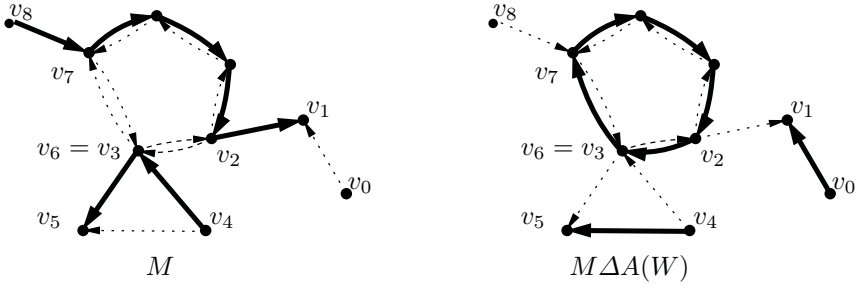


Fig. 1. A special M-alternating walk $W = (\ _0, \cdots, \ _8)$ with $\ = \ _0$ and the symmetric difference $M \ \ A(W)$

Notice, that $M\Delta A(W)$ is not necessarily an even factor, see Figure 1 for example. On the left hand side of Figure 1 we have an odd-cycle-symmetric graph where the bold lines form an even factor M, and $v_0, v_1, \ldots, v_8$ is a special M-alternating walk. On the right hand side the bold lines form a path-cycle-factor $M\Delta A(W)$, illustrating Lemma 12.

Proof of Theorem 1. We prove Theorem 1 by induction on $|V|$. Consider an odd-cycle-symmetric digraph $D = (V, A)$, let $M \subseteq A$ be a maximum even factor. We need to show that there is a verifying set X giving equality $|M| = \tau_D(X)$. Recall the definition of $V^-(M) := \{z \in V : \varrho_M(z) = 1\}$ which is the set of M-source-nodes.

Case I. Suppose there is an arc $ab = e \in A$ with $a \in L = L(D, M)$ and $b \in V - V^-(M)$. In this case we will find a maximum even factor N which is nice with an odd cycle, the proof will be completed using Theorem 2. By Lemma 11 there is a special even M-alternating walk W with last node a, suppose $q = v_0 \in K^+$ is the first node, i.e.

$$W = (q = v_0, e_0, v_1, e_1, \ldots, v_{n-1}, e_{n-1}, v_n = a).$$

Let $n = 2l$ be the length of W, let W_i be the starting segments of W length $2i$ (for $i = 0, \ldots, l$), of course each W_i is a special even M-alternating walk. By Lemma 12 $M_i := M\Delta A(W_i)$ are path-cycle-factors, here $M_0 = M$. It is easy to see that for $i \leq n - 1$

$$M_{i+1} = M_i + v_{2i}v_{2i+1} - v_{2i+2}v_{2i+1}, \tag{8}$$

i.e. M_{i+1} is obtained from M_i by replacing an arc entering v_{2i+1} by a different arc entering v_{2i+1}. (Figure 2 illustrates the sequence of M_i's for the example of D, W given in Figure 1.)

Subcase Ia. Suppose M_l is an even factor. It is easy to see that M_l has no arc leaving a; by assumption $b \in V - V^-(M)$, thus M_l has no arc entering b; hence $M_l + ab$ is a path-cycle-factor. $M_l + ab$ is not an even factor, since $|M_l + ab| > |M|$. As M_l is an even factor, there must be a unique odd cycle $C \subseteq M_l + ab$, and for this cycle $ab \in C$. So $N := M_l$ is nice with C.

Subcase Ib. Suppose M_l is not an even factor. M_0 is an even factor, consider the smallest $0 \leq i < l$ for which M_i is an even factor, but M_{i+1} is not an even factor. Then by (8) there is a unique odd cycle C in M_{i+1}, it is easy to see that it contains $v_{2i}v_{2i+1}$, so $N := M_i$ is nice with C. (In Figure 2 M_3 is nice with a cycle of length 5 occurring in M_4.)

Now we have a maximum even factor N which is nice with an odd cycle C. By part 1 of Theorem 2 $N' = N/V(C)$ is a maximum even factor in $D' = D/V(C)$. By induction, there is a verifying set X' for N' in D'. By part 2 of Theorem 2 there is a verifying set for N in D.

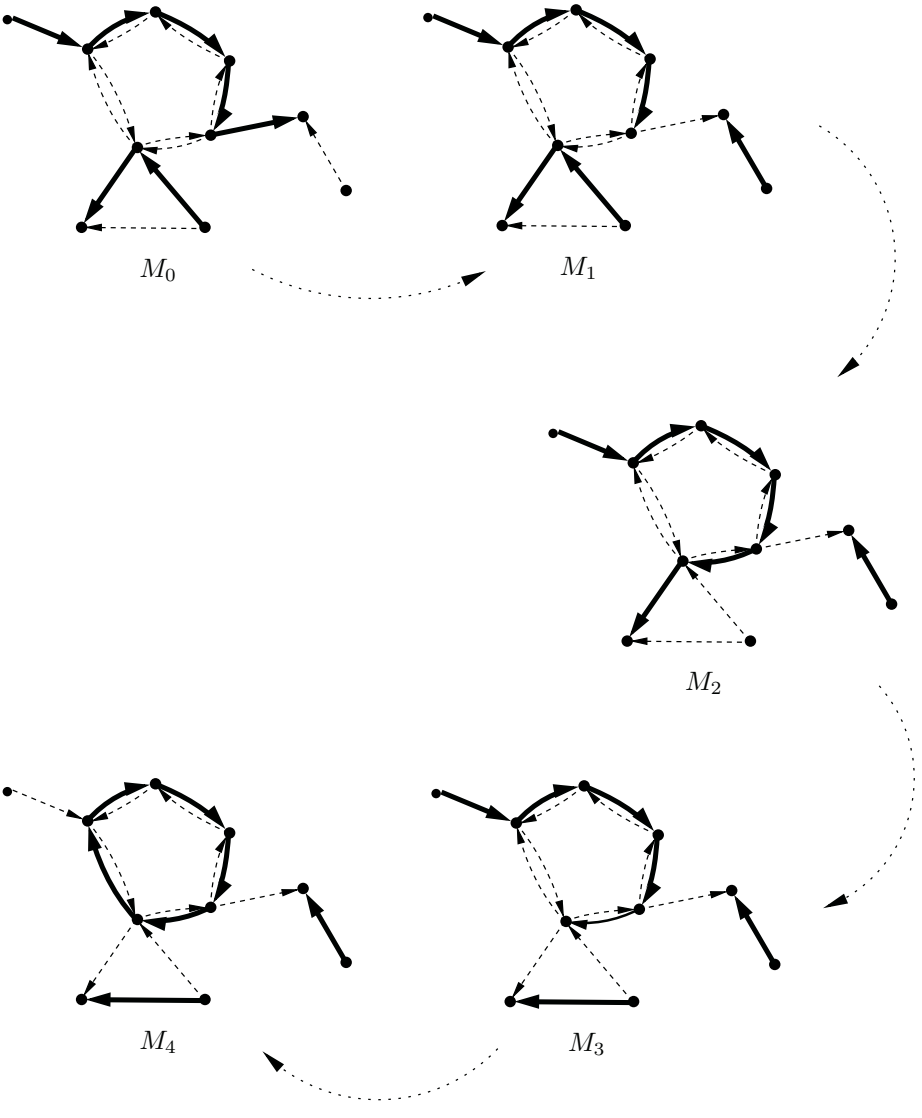


Fig. 2. The sequence of M_i's in Case I for W indicated in Figure 1

Case II. Suppose there is no arc $ab = e \in A$ with $a \in L = L(D, M)$ and $b \in V - V^-(M)$.

We will prove that L is a verifying set. Let $M_1 := M[L] = \{vz \in M : v \in L, z \in L\}$, $M_2 := \{vz \in M : v \in L, z \in V - L\}$ and $M_3 := \{vz \in M : v \in V - L\}$. Let S be the set of source-nodes in $D[L]$.

Lemma 13. *In Case II we have* $|M_3| = |V| - |L|$, $|M_2| = |\Gamma_D^+(L)|$ *and* $|M_1| = |L| - |S|$.

Proof. The first equality follows from $K^+ \subseteq L$.

Consider a node b in $\Gamma_D^+(L)$, then there must be a node a in L for which $ab \in A$. $b \in V^-(M)$ follows from the assumption of case II, thus there must be an arc $cb \in M$. By the definition of L there is an even M-alternating walk W with last node a, the extension of W by arcs ab and cb gives an even M-alternating walk with last node c, thus $c \in L$. We get that each node in $b \in \Gamma_D^+(L)$ is covered by an arc $cb \in M_2$ with $c \in L$, which implies the second equality.

For the third equality, consider a node $b \in L - S$; then there is an arc $ab \in A$ with $a \in L$. From the assumption of case II we get that b must be in $V^-(M)$, i.e. there must be an arc $cb \in M$. There is an even M-alternating walk W with last node a, the extension of W by arcs ab and cb gives an even M-alternating walk with last node c, thus $c \in L$. We get that each node in $b \in L - S$ is covered by an arc $cb \in M_3$, which implies the third equality.

The proof of Theorem 1 is completed by the following calculation:

$$|M| = |M_1| + |M_2| + |M_3| = |V| + |\Gamma_D^+(L)| - |S| \geq |V| + |\Gamma_D^+(L)| - \sigma(D[L]) \geq |M|.$$

4 A Gallai-Edmonds-Type Structure

A structural description was given by Edmonds [6] and Gallai [7] on the structure of maximum matchings in a graph. The properties of the Gallai-Edmonds decomposition are extended to even factors in the following theorem, a proof was given in [12] for weakly symmetric digraphs, which also works for odd-cycle-symmetric digraphs. Here we give a different proof.

Theorem 3 (Pap, Szegő, [12]). *Suppose* $D = (V, E)$ *is odd-cycle-symmetric, let* L_D *be the set of nodes* $v \in V$ *for which there is a maximum even factor* M *with* $\delta_M(v) = 0$. *Then the following assertions hold.*

1. $\nu(D) = |V| + |\Gamma_D^+(L_D)| - \sigma(D[L_D])$
2. *For any verifying set* X *we have* $L_D \subseteq X$.
3. *All source-component in* $D[L_D]$ *is symmetric-critical.*
4. M *is a maximum even factor if and only if it fulfills the inequalities (5)–(7) for* $X = L_D$ *with equality.*

Proof. Assertion 2 also follows from assertion 1 by Lemma 4, assertion 3 follows from assertion 2, assertion 4 also follows from assertion 1. We prove the first assertion by induction on $|V|$. Consider a maximum even factor M, in Section 3 we have shown that (Case I) there is a maximum even factor N which fits an odd cycle C or (Case II) $L(D, M)$ is a verifying set.

First suppose $L(D, M)$ is a verifying set. By Lemma 4 $L_D \subseteq L(D, M)$, if $L_D = L(D, M)$ then we are done. Otherwise consider a node $v \in L(D, M) - L_D$,

by definition there is a special even M-alternating walk W with last node v. $M\Delta A(W)$ is a path-cycle-factor with no arc leaving v, so by $v \notin L_D$ we get that $M\Delta A(W)$ is no even factor. By the argument in Subcase Ib we get a maximum even factor N which fits an odd cycle C.

If a maximum even factor N fits an odd cycle C, then by Theorem 2 $N' = N/V(C)$ is a maximum even factor in $D' = D/V(C)$. Thus $\{C\} \in L_{D'}$. By induction $L_{D'}$ is a verifying set in D', let X be its pre-image in D. We need to show that X is a verifying set and $X \subseteq L_D$, since by Lemma 4 these imply $X = L_D$.

N is a maximum even factor fitting C, thus the nodes in $V(C)$ are easily seen to be in L_D. Consider a node $v \in X - \{C\}$, let M be a maximum even factor in D' with $\delta_M(v) = 0$. The construction in Lemma 9 gives a maximum even factor $M'' := M' \cup M_{st}$ in D with $\delta_{M''}(v) = 0$, this proves $X \subseteq L_D$.

Lemma 10 implies that X is a verifying set.

Remark. In Theorem 3 the set L_D corresponds to the set "D", while $\Gamma_D^+(L_D)$ corresponds to the set "A" in the Gallai-Edmonds decomposition. The change of notation is to avoid ambiguity of the notation D for the digraph.

5 Conclusions and Examples

1.) Let us try to distinguish concepts which are basically the same as for the matching case, and concepts which needed to be modified – compared to Edmonds' matching algorithm. The following theorem is the analogue of Theorem 2 for maximum matchings.

Theorem 4. *Suppose N is a matching in an undirected graph G and C is an N-alternating odd cycle adjacent to an exposed node; define $G' = G/V(C)$ and $N' = N/V(C)$.*

1. *If N is maximum, then N' is maximum.*
2. *If Z' is a barrier (in the Berge-Tutte formula) for N', then the pre-image Z is a barrier for N.*

In fact the definition of "N fitting C" was motivated by the nice correspondence of Theorems 4 and 2. An odd cycle C with the above properties can be used in an inductive or algorithmic way – both for matchings and for even factors. The key difference between the concept of this paper and well-known concepts for matchings is the way how we find this odd cycle.

2.) Notice that M-alternating walks are easier to handle than alternating paths: the concatenation of two M-alternating walks is an M-alternating walk, too. The analogue statement neither holds for M-alternating paths for matchings, nor for special walks for even factors.

For this reason, the proof and the algorithm uses M-alternating walks, let us sketch how this works for the special case of matchings. The algorithm first tries

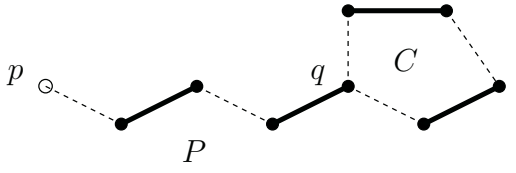


Fig. 3. An M-alternating blossom

to find an alternating walk W joining two exposed nodes, consider a shortest one. If W is an alternating path, then the actual matching can be augmented. Otherwise, let W' be a shortest segment of W which is not a path. One can easily see that W' is an M-alternating blossom, i.e. $W' = P \cup C$ where P is an M-alternating even path from an exposed node p to q, and C is an M-alternating odd cycle for which $V(P) \cap V(C) = \{q\}$ (see Figure 3). We apply Theorem 4 for $M\Delta P$, C, we are done by induction.

If there is no M-alternating walk W joining two exposed nodes, then the set $L := L(\overleftrightarrow{G}, \overleftrightarrow{M})$ is an indepedent set, and $\Gamma_{\overleftrightarrow{G}}(L)$ will be a verifying set in the Berge-Tutte formula.

3.) Consider the reduction of the maximum matching problem in a graph G to the maximum even factor problem in $\overleftrightarrow{G}$. A maximum matching in G has exactly $\nu(\overleftrightarrow{G})/2$ edges, since (a) for a matching M in G we can give the even factor $\overleftrightarrow{M}$ having two-edge cycles in place of the edges of M, and (b) if M is an even factor in $\overleftrightarrow{G}$, then an even cycle in M can be replaced by a matching with half as many edges, a directed path of length l can be replaced by a matching with $\lceil l/2 \rceil$ edges.

The proof of Theorem 1 can be specialized to the instance $\overleftrightarrow{G}$ to prove the Berge-Tutte formula, but we cannot avoid directed paths to occur. Hence, we in fact solve the "maximum 2-matching without odd cycles" problem.

A 2-matching is a node-disjoint family of edges with weight 2, and paths or cycles with weight 1 on each edge. In Figures 4, 5 only the supporting edge sets of 2-matchings are indicated with bold lines. Alternating walks, the symmetric difference should be understood by the reduction to $\overleftrightarrow{G}$, an alternating walk is

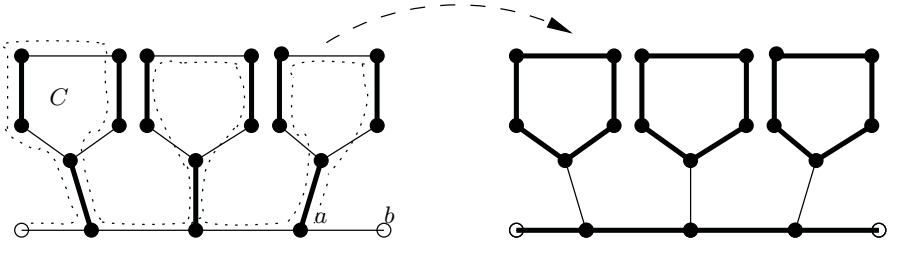


Fig. 4.

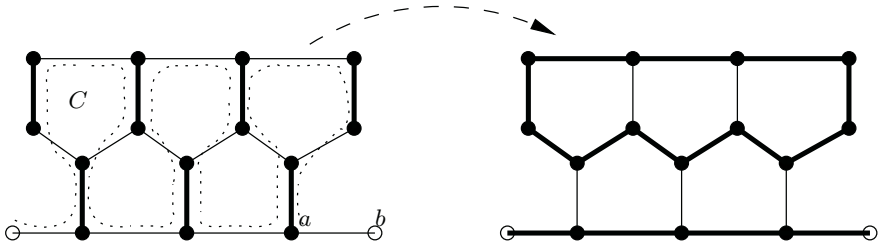


Fig. 5.

indicated by a dotted line on the left hand side, while the symmetric difference is indicated on the right hand side.

We explain Subcase Ib by the example shown in Figure 4. q, b are the exposed nodes, the dotted line is a special even M-alternating walk W from q to a, the edge ab has the role of Case I. The algorithm constructs the sequence of M_i's step by step with i increasing. When the algorithm gets to $i = 4$ an odd cycle C occurs which will be contracted.

In Subcase Ib, $M_l + ab$ is not an even factor, so it has at least one odd cycle. In general, $M_l + ab$ may have several odd cycles in this case; for example in Figure 4 we have $l = 12$ and the 2-matching $M_{12} + ab$ has three odd cycles. The algorithm does not choose one of them as C for the contraction, Figure 5 (using the same notation) shows an example where the cycle C does not appear in $M_l + ab$.

Acknowledgement

The author is grateful to András Frank for his helpful suggestions.

References

1. E. Balas and W. Pulleyblank, *The perfectly matchable subgraph polytope of an arbitrary graph*, Combinatorica **9** (1989) 321-337.
2. C. Berge, *Sur le couplage maximum d'un graph*, Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences **247** (1958) 258-259.
3. W.H. Cunningham, *Matching, Matroids and Extensions*, Math. Program. Ser. B **91** (2002) 3, 515-542.
4. W.H. Cunningham and J.F. Geelen, *The Optimal Path-Matching Problem*, Combinatorica, **17/3** (1997), 315-336.
5. W.H. Cunningham and J.F. Geelen, *Combinatorial Algorithms for Path–Matching*, manuscript, (2000).
6. J. Edmonds, *Paths, trees, and flowers*, Canadian Journal of Mathematics **17** (1965) 449-467.
7. T. Gallai, *Maximale Systeme unabhängiger Kanten*, A Magyar Tudományos Akadémia Matematika Kutatóintézetének Közleményei **9** (1964) 401-413.
8. A. Frank and L. Szegő, *A Note on the Path-Matching Formula*, J. of Graph Theory **41/2**, (2002) 110-119.

9. L. Lovász, *On determinants, matchings and random algorithms*, in Fundamentals of Computational Theory (L. Budach, ed.), Akademie-Verlag, Berlin, 1979, 565-574.
10. L. Lovász and M.D. Plummer, Matching Theory, Akadémiai Kiadó, Budapest, 1986.
11. T. Király and M. Makai, *On polyhedra related to even factors*, EGRES Technical Report TR-2003-09
12. G. Pap and L. Szegő, *On the Maximum Even Factor in Weakly Symmetric Graphs*, Journal of Combinatorial Theory Ser. B, **91/2** (2004) 201-213.
13. B. Spille and L. Szegő, *A Gallai-Edmonds-type Structure Theorem for Path-Matchings*, Journal of Graph Theory, **46/2** (2004) 93-102.
14. B. Spille and R. Weismantel, *A Generalization of Edmonds' matching and matroid intersection algorithms*, in: Proceedings of the 9th International IPCO Conference, 2002; W.J. Cook, A.S. Schultz, eds.
15. W.T. Tutte, *The factorization of linear graphs*, The Journal of the London Mathematical Society **22**, (1947) 107-111.

Improved Approximation Schemes for Linear Programming Relaxations of Combinatorial Optimization Problems

Fabián A. Chudak and Vânia Eleutério

Institute for Operations Research,
Swiss Federal Institute of Technology, ETH Zürich, Switzerland
{chudak, eleuterio}@ifor.math.ethz.ch
http://www.ifor.math.ethz.ch

Abstract. We consider a generic paradigm to design improved PTAS for linear programming relaxations of combinatorial optimization problems. For the case of the uncapacitated facility location problem, the scheduling problem $R||C_{\max}$ and the set covering problem we substantially improve the running time dependence on from the previously known $O(1/\ ^2)$ to $O(1/\ )$. All these algorithms are remarkably simple to implement. For the survivable network design problem we improve the dependence from $O(1/\ ^2)$ to $O((1/\ )\log(1/\ ))$. We present some preliminary computational results that seem to suggest that our algorithms may prove very competitive in practice. Furthermore, we present a general approximation version of a result of Nesterov that potentially can be applied to a very large class of linear programming problems.

Our results build mainly on work of Nesterov and extend the work of Bienstock and Iyengar. In fact, one of the objectives of this paper is to make clearer the relationship between these two.

1 Introduction

In this paper we present algorithms for approximately solving special cases of linear programming problems. All the problems we study are linear programs of polynomial size and, thus, can be solved using standard algorithms (e.g., interior point methods) in polynomial time. However, from a practical point of view, very often large scale instances of these linear programs cannot be handled using these algorithms. In these circumstances, practitioners have used decomposition methods extensively. These methods typically consider some form of Lagrangian relaxation followed by some variant of subgradient optimization. From a theoretical point of view, the question of validating this type of decomposition approaches has been translated into different types of convergence analysis. Among these, researchers have studied the worst-case convergence rate of the algorithms. This paper fits in this line of research.

For a minimization linear program (LP) we are interested in finding a polynomial time approximation scheme (PTAS). In other words, given an $\epsilon > 0$

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 81–96, 2005.

our task is to find a feasible solution within a factor of $1+\epsilon$ of the LP optimum (which we always denote by OPT) in polynomial time. The focus of this paper is on the dependence of the running time on ϵ. Up to recently algorithms for approximately solving LP's have typically considered a Lagrangian relaxation followed by something that can be usually described as a subgradient optimization procedure. The literature is extensive. The book of Bienstock [5] covers many of these algorithms for multicommodity flow problems and some of its generalizations; we also refer the reader to [10, 11, 26, 9]. All of these algorithms have a running time dependence on ϵ of $O(1/\epsilon^2)$. On the negative side, there is some compelling evidence that algorithms that are only allowed to obtain a subgradient (in addition to function evaluation) at each iteration require $\Omega(1/\epsilon^2)$ iterations (see [20, 14]). These results do not completely rule out the possibility of a faster subgradient-based algorithm, but do suggest that one needs to use substantial additional information from the problem structure.

In a departure from subgradient-optimization based algorithms, Nesterov [21, 19] proposed a method that uses gradient optimization instead. The method can be essentially explained as follows. There are two steps: smoothing and gradient optimization. In the smoothing step the idea is to make the Lagrangian function differentiable. In the second step we apply an "optimal" gradient optimization algorithm (see [20]) which has a convergence rate of $O(L/\sqrt{\epsilon})$, where L is the Lipschitz continuous constant of the gradients of the function. The smoothing process can be tailored to produce a function whose gradient Lipschitz continuous constants grow as $O(A/\sqrt{\epsilon})$. Thus for an additive error of ϵ, the required number of iterations is $O(A/\sqrt{\epsilon})/\sqrt{\epsilon} = O(A/\epsilon)$. There are two caveats: (a) the dependence on the data, namely A, usually is not polynomial, and (b) the subproblems that need to be solved at each iteration are non-linear and can be very non-trivial (even harder than the original problem). Bienstock and Iyengar [6] were the first to adapt the methods of [21] to design PTAS for the maximum concurrent multicommodity flow problem with a running time dependence on ϵ of $O(1/\epsilon \log(1/\epsilon))$ improving the previously known $O(1/\epsilon^2)$. The first "pure" $1/\epsilon$ dependence on ϵ that we are aware of are in [23, 22] which were developed at about the same time as our algorithms (in the packing problem of [6], the issue of solving the subproblems is not addressed in detail).

In this paper we explore further applications and extensions of the work of [21, 19]. Our approach is very different from that of [6]. In [6] their focus was on extending previous work incorporating only the gradient optimization part of the approach of [21]. In effect, they only considered one type of smoothing that was used frequently in the literature [5, 26]: the logarithmic approximation of the max ($\max_i x_i \approx \mu \ln(\sum_i e^{x_i/\mu})$). In contrast, in this paper we use the theory of Nesterov at a fuller extent. We show that to deal with problem (a) above we can restrict the feasible region using additional information about OPT and using binary search (this method has been used before [5, 26, 10, 6]). For dealing with the subproblems, that is (b) above, we show first that in some important special

cases they are easy to solve, and second that an approximate version of the results of Nesterov can accommodate approximate solutions to the subproblems extending the work of [6].

Our main contributions are the following. We present improved PTAS for the linear programming relaxation of the uncapacitated facility location problem, the scheduling problem $R||C_{\max}$ [1] , the set covering problem and the survivable network design problem. All of our algorithms find feasible solutions that are within $(1+\epsilon)$ of the fractional optimum and run in time proportional to $1/\epsilon$ for the first three problems mentioned above and $(1/\epsilon)\log(1/\epsilon)$ for the survivable network design problem. Our algorithms improve the previously known dependences on ϵ of $O(1/\epsilon^2)$ at a moderate increase in the dependence on the polynomial on the size of the input. Furthermore we elaborate on a generic paradigm to design improved PTAS that potentially could be applied to other linear programming problems. Many of our algorithms can be easily implemented and we report some preliminary computational experience that suggests that they could be very competitive in practical applications.

In Section 2, we elaborate on our methods for the uncapacitated facility location problem. In Section 3, we summarize the steps we needed to design our algorithms, revisit the results of [6], and present algorithms for the scheduling problem $R||C_{\max}$ and the set covering problem. In Section 4, we present our approximate version of the results of Nesterov, an algorithm for the survivable network design problem and a discussion on further applications. Finally Section 5 reports on preliminary computational results.

2 The Algorithm for the Uncapacitated Facility Location Problem

In the uncapacitated facility location problem (UCFLP) there is a set of potential facility locations $\mathcal{F}$; building a facility at location $i \in \mathcal{F}$ has an associated fixed cost $f_i \geq 0$, and any open facility can provide an unlimited amount of a certain commodity. There is also a set of clients $\mathcal{D}$ that require service; client $j \in \mathcal{D}$ has a demand $d_j > 0$ that must be shipped from one of the open facilities. If a facility at location $i \in \mathcal{F}$ is used to satisfy the demand of client $j \in \mathcal{D}$, the service cost incurred is proportional to the distance from i to j, c_{ij}. The goal is to determine a subset of the set of potential facility locations at which to open facilities and an assignment of clients to these facilities so as to minimize the overall total cost, that is, the fixed costs of opening the facilities plus the total service cost. We assume that all the demands are 1 (otherwise we replace c_{ij} with $d_j c_{ij}$). Throughout we let $m = |\mathcal{F}|$ and $n = |\mathcal{D}|$. The sim-

[1] This is a special case of packing linear programs considered in [6]. However no general run-time bound is provided. Even assuming that their approach to the maximum concurrent multicommodity flow problem can be extended, our algorithm is at least a factor of $O(\log(1/\ )\sqrt{\ \ })$ faster and much simpler.

plest linear programming relaxation (from [2]), which we will refer to as P, is as follows:

$$\text{Minimize} \quad p(x,y) = \sum_{j\in\mathcal{D}}\sum_{i\in\mathcal{F}} c_{ij}x_{ij} + \sum_{i\in\mathcal{F}} f_i y_i$$

$$\begin{aligned} \text{subject to} \quad & \sum_{i\in\mathcal{F}} x_{ij} = 1 \qquad \forall j \in \mathcal{D} && (1)\\ & x_{ij} \le y_i \qquad \forall j \in \mathcal{D}, i \in \mathcal{F} && (2)\\ & x_{ij} \ge 0. \end{aligned}$$

Any 0-1 feasible solution corresponds to a feasible solution to the UCFLP: $y_i = 1$ indicates that a facility at location $i \in \mathcal{F}$ is open, whereas $x_{ij} = 1$ means that client $j \in \mathcal{D}$ is serviced by the facility built at location $i \in \mathcal{F}$. Inequalities (1) state that each demand point $j \in \mathcal{D}$ must be assigned to some facility, whereas inequalities (2) say that clients can only be assigned to open facilities. Thus the linear program P is indeed a relaxation of the problem. The linear programming relaxation P is known to provide excellent lower bounds, and sometimes is referred to as the "strong linear programming relaxation" (see [18] for a practical point of view, and [8] and references thereafter for a theoretical point of view). Because of its large size ($O(mn)$ variables and constraints), Lagrangian relaxations have been considered to solve it. In practice, the algorithms that are known to obtain the best results are those that relax equations (1). However these algorithms usually have problems delivering feasible primal solutions (see [4]) and do not have a known worst-case analysis of the running time. In contrast, from a theoretical point of view, relaxing constraints (2) has proved more useful. In this category are the algorithms of [25], [10] and [7]. In this paper, we follow this approach as well.

Next we proceed to describe an algorithm that has an additive error of ϵ. To simplify the notation, we will use Δ_m to denote the m-dimensional simplex, that is, $\Delta_m = \{z \in \mathbb{R}^m : \sum_{i=1}^m z_i = 1, z_i \ge 0, \text{ for } i = 1,\ldots,m\}$. Thus, if (x,y) is feasible for P, then $x \in \Delta_m^n$. If now we let w_{ij} be the multipliers of inequalities (2), we have the following Lagrangian formulation of P: $\max_{w_{ij}\ge 0}\{\min_{x\in\Delta_m^n} \sum_{i=1}^m\sum_{j=1}^n (c_{ij}+w_{ij})x_{ij} + \sum_{i=1}^m y_i(f_i - \sum_{j=1}^n w_{ij})\}$. Since we do not require that the y_i's be nonnegative, we can assume that for each $i = 1,\ldots,m$, $f_i = \sum_{j=1}^n w_{ij}$. If now we use the change of variables $f_i u_{ij} = w_{ij}$, we get the following representation of P: $\max_{u\in\Delta_n^m}\{\min_{x\in\Delta_m^n} \sum_{i=1}^m\sum_{j=1}^n c_{ij}x_{ij} + \sum_{i=1}^m\sum_{j=1}^n f_i u_{ij} x_{ij}\}$. For fixed $\overline{u} \in \Delta_n^m$ and $\overline{x} \in \Delta_m^n$ consider the following two functions

$$d(\overline{u}) = \min_{x\in\Delta_m^n} \Big\{ \sum_{i=1}^m\sum_{j=1}^n c_{ij}x_{ij} + \sum_{i=1}^m\sum_{j=1}^n f_i\overline{u}_{ij}x_{ij}\Big\},$$

$$p(\overline{x}) = \max_{u\in\Delta_n^m} \{\sum_{i=1}^m\sum_{j=1}^n c_{ij}\overline{x}_{ij} + \sum_{i=1}^m\sum_{j=1}^n f_i u_{ij}\overline{x}_{ij}\}.$$

From the argument above, we know that $\max_{u\in\Delta_n^m} d(u)$ has the same optimal value as P. The same is true for $\min_{x\in\Delta_m^n} p(x)$ as we show next, and indeed these two optimization problems map exactly to the standard primal and dual formulations of P. For fixed $x \in \Delta_m^n$, observe that since the $f_i \geq 0$, we can always consider the feasible solutions (x, y) with $y_i = \max_{j=1,\ldots,n} x_{ij}$ $(i = 1, \ldots, m)$. Thus the equivalence of P and $\min_{x\in\Delta_m^n} p(x)$ follows immediately from the elementary fact that for fixed i, $\max_{j=1,\ldots,n} x_{ij} = \max_{u_i\in\Delta_m} \sum_{j=1}^n u_{ij}x_{ij}$ (where $u_i \in \mathbb{R}^n$, $(u_i)_j = u_{ij}$).

Next we state a result of Nesterov [19] in some generality that we will need later. Suppose that $Q_1 \subseteq \mathbb{R}^{\overline{n}}$, $Q_2 \subseteq \mathbb{R}^{\overline{m}}$ are bounded, closed, and convex sets and $A \in \mathbb{R}^{\overline{m}\times\overline{n}}$, $c \in \mathbb{R}^{\overline{n}}$ and $b \in \mathbb{R}^{\overline{m}}$. Consider the functions $f(x) = c^t x + \max_{u\in Q_2}\{\langle Ax, u\rangle - b^t u\}$ $(x \in Q_1)$ and $\phi(u) = -b^t u + \min_{x\in Q_1}\{c^t x + \langle Ax, u\rangle\}$ $(u \in Q_2)$, where $\langle Ax, u\rangle = (Ax)^t u$. Note that $f(x) \geq \phi(u)$ (weak-duality). The goal is to solve $\min_{x\in Q_1} f(x)$ and $\max_{u\in Q_2} \phi(u)$. Suppose next that we fix the norms $||\ ||_{①}$ in $\mathbb{R}^{\overline{n}}$ and $||\ ||_{②}$ in $\mathbb{R}^{\overline{m}}$ (index ① indicates the space of the x's in $\mathbb{R}^{\overline{n}}$ and index ② indicates the space of the u's in $\mathbb{R}^{\overline{m}}$; they can be *any* norms). Now we can define the operator norm of A as $||A||_{①,②} = \max\{\langle Ax, u\rangle : ||x||_{①}, ||u||_{②} \leq 1\}$. In addition, we fix two strongly convex functions d_1 defined on $\mathbb{R}^{\overline{n}}$ and d_2 defined on $\mathbb{R}^{\overline{m}}$. Let $x_\circ = \arg\min_{x\in Q_1} d_1(x)$ and $u_\circ = \arg\min_{u\in Q_2} d_2(u)$, without loss of generality $d_1(x_\circ) = d_2(u_\circ) = 0$. We assume that the strongly convex constants with respect to the chosen norms are respectively σ_1 and σ_2, so that $d_1(x) \geq \frac{1}{2}\sigma_1||x - x_\circ||^2_{①}$ for all $x \in Q_1$ and $d_2(u) \geq \frac{1}{2}\sigma_2||u - u_\circ||^2_{②}$ for all $u \in Q_2$. Let $D_1 = \max_{x\in Q_1} d_1(x)$ and $D_2 = \max_{u\in Q_2} d_2(u)$.

Theorem 1 ([19]). *There is a primal-dual algorithm that finds a sequence of points $(x_k, u_k) \in Q_1 \times Q_2$ such that the duality gap of the N^{th} iteration is*

$$f(x_N) - \phi(u_N) \leq \frac{4||A||_{①,②}}{N+1}\sqrt{\frac{D_1 D_2}{\sigma_1\sigma_2}}. \tag{3}$$

At each iteration the algorithm needs to compute the exact solution of three problems of the form $\min_{x\in Q_1} \alpha^t x + d_1(x)$ $(\alpha \in \mathbb{R}^{\overline{n}})$ or $\min_{u\in Q_2} \beta^t u + d_2(u)$ $(\beta \in \mathbb{R}^{\overline{m}})$. Alternative variants need the exact solution of problems of the form $\min_{z\in Q_i} \gamma^t z + ||z - \overline{z}||^2_{①}$ $(\gamma \in \mathbb{R}^{\overline{n}}$ or $\gamma \in \mathbb{R}^{\overline{m}})$, $\overline{z} \in Q_i$ fixed, $i = 1, 2$.

The proof of the theorem is based on making both functions f and ϕ differentiable by perturbing the max (respectively min) with the strongly convex function d_2 (resp. d_1). The perturbation makes the solution of the max (resp. min) problems unique and, thus, differentiable. The gradients of these perturbations give good directions of improvement and are used iteratively to reduce the duality gap. The proofs of convergence are simple and ingenious, and the reader is referred to [19]. A very nice property of the algorithms of the theorem that all of our algorithms inherit is that if our target is an absolute error of $\epsilon > 0$, the algorithm does not use ϵ during the computations; ϵ is needed solely to determine how many iterations to run the algorithm to make sure that the bound (3) holds.

We apply Theorem 1 to the UCFLP as follows. The functions p and d correspond with f and ϕ, the set Q_1 is Δ_m^n, the set Q_2 is Δ_n^m, the operator A is

defined by $\langle Ax, u\rangle = \sum_{i=1}^{m}\sum_{j=1}^{n} f_i u_{ij} x_{ij}$, c is our cost vector and $b = 0$. We choose now the standard Euclidean norm in both spaces x and u. Next we set $d_1(x) = \frac{1}{2}||x - x_\circ||^2_{①}$, where $(x_\circ)_{ij} = \frac{1}{m}$ $(i = 1, \ldots, m,\ j = 1, \ldots, n)$, that is, the j^{th} column of $x_\circ$ is the center of the j^{th} simplex of the feasible primal region Δ_m^n, and similarly, $d_2(u) = \frac{1}{2}||u - u_\circ||^2_{②}$, where $(u_\circ)_{ij} = \frac{1}{n}$ $(i = 1, \ldots, m,\ j = 1, \ldots, n)$. Thus we have that $\sigma_1 = \sigma_2 = 1$. Notice that for $x \in \Delta_m^n$, $\frac{1}{2}||x - x_\circ||^2_{①} = \frac{1}{2}(||x||^2_{①} - \frac{n}{m})$, and thus its maximum value is $D_1 = \frac{1}{2}n\frac{m-1}{m}$. Similarly, if $u \in \Delta_n^m$, the maximum of $\frac{1}{2}||u - u_\circ||^2_{②}$ is $D_2 = \frac{1}{2}m\frac{n-1}{n}$. Let $F = \max_{i=1}^{n} f_i$, that is, the maximum facility cost. Finally we need to compute $||A||_{①,②}$, but this is simple too since $\sum_{i,j} f_i |x_{ij} u_{ij}| \le F \sum_{i,j} |x_{ij} u_{ij}| \le F||x||_{①}||u||_{②}$ (using Cauchy-Schwarz), and thus $||A||_{①,②} \le F$. With all these ingredients, plugging in (3) we have the following.

Theorem 2. *Given $\epsilon > 0$, if we run algorithm corresponding to Theorem 1 $N = \lceil \frac{1}{\epsilon} 2F\sqrt{(n-1)(m-1)} - 1 \rceil$ iterations, we obtain primal and dual feasible solutions, $x \in \Delta_m^n$ and $u \in \Delta_n^m$, such that $p(x) - d(u) \le \epsilon$.*

Two issues need to be addressed carefully: the run time of each iteration and the dependence on the facility costs F. We address first the former. Focus in one iteration of the algorithm of Theorem 2. According to Theorem 1 each iteration needs to solve three optimization problems. In our case, all of the subproblems take the form $\min_{x \in \Delta_m^n} \left\{ \langle \alpha, x\rangle + ||x - \overline{x}||^2_{①} \right\}$ (primal) or $\min_{u \in \Delta_n^m} \left\{ \langle \beta, u\rangle + ||u - \overline{u}||^2_{②} \right\}$ (dual) for some fixed $\alpha, \beta \in \mathbb{R}^{mn}$ and $\overline{x} \in \Delta_m^n$, $\overline{u} \in \Delta_n^m$. By symmetry, we will only consider the primal problems. First of all, notice that by changing α, the primal problem can be written as $\min_{x \in \Delta_m^n} \{\langle \alpha, x\rangle + \frac{1}{2}\sum_{ij} x_{ij}^2\}$. And thus, it is separable on the clients, that is, the minimum can be written as the sum $\sum_j \min_{x_j \in \Delta_m} \{\sum_i \alpha_{ij} x_{ij} + \frac{1}{2}\sum_i x_{ij}^2\}$. Thus we only need to show how to solve problems of the form $\min_{z \in \Delta_m} \left\{ \frac{1}{2}\sum_{i=1}^{m} z_i^2 + \sum_{i=1}^{m} a_i z_i \right\}$, for any $a \in \mathbb{R}^m$. We show next that this quadratic convex constrained problem can be solved very efficiently as follows. First we sort the a_i's and assume that $a_1 \le a_2 \le \ldots \le a_m$; in addition, since $\sum_i z_i = 1$, we can also assume that $a_1 = 0$. Since the objective function is strictly convex and the feasible region is convex, the optimal solution, z^*, is unique. Also, it is easy to see that $z_1^* > 0$ (if $z_1^* = 0$ and any other $z_j^* > 0$ with $a_j > 0$, we can interchange z_j^* and z_1^*; if all the positive z_j^* have $a_j = 0$, the optimal solution is to split 1 evenly among those). Using the KKT conditions, it is simple to argue that in an optimal solution $z_i^* = (z_1^* - a_i)^+$ for $i = 1, \ldots, m$. Finally to set z_1^* we use the equation $\sum_i z_i^* = 1$. The whole procedure can be implemented in $O(m \log m)$ time. We also point out that z^* is the unique optimal solution of the following LP (which we will refer as SP): $\max\{z_1 : \sum_{i=1}^{m} z_i \le 1, z_1 \le z_i + a_i\ (i = 1, \ldots, m)\ , z_i \ge 0\ (i = 1, \ldots, m)\}$. Thus, we can give the following interpretation. Suppose that this subproblem came from client j so that client j has to decide the proportion of how much to assign to each facility. In a subgradient algorithm, it will simply choose one facility, the one that it perceives as the cheapest ($\arg\min_i a_i$). Instead, in our algorithm, we first do a normalization step (sorting and setting $a_1 = 0$) and then do the assignment by

solving SP. We still want to assign j to facility 1 (the cheapest), if we imagine that z_1 starts growing from 0, each time that our assigned proportion z_1 covers facility i (i.e., $z_1 \geq a_i$) we must start covering facility i as well (start increasing z_i). Once all of j is allocated, we stop. The dual of SP has also a combinatorial interpretation: we want assign j to facilities, the cost of assigning j to facility i is a_i, the facility costs are 1, but all open facilities have to be assigned the same fraction. The optimal solution of this problem chooses the k cheapest facilities (for some unique k) and assigns j evenly among them. From a theoretical point of view, sorting can be avoided altogether using a variant of the weighted median algorithm [2] (see [15]), thus obtaining a linear time algorithm.

Lemma 1. *All the subproblems of the algorithm of Theorem 2 are separable by client or facility and can be solved in $O(nm)$ time each.*

Next we show how to transform Theorem 2 into a polynomial time approximation algorithm. The following two results closely follow [10].

Lemma 2. *Let $(\overline{x}, \overline{y})$ be any feasible solution of* P*. Then if some $\overline{y}_{i_\circ} > 0$ with $f_{i_\circ} > p(\overline{x}, \overline{y})$, we can find a new solution $(\widehat{x}, \widehat{y})$ with strictly smaller objective function value such that $\widehat{y}_{i_\circ} = 0$.*

Proof. Note first that it must be that $\overline{y}_\circ < 1$. Define $\widehat{x}_{ij} = \overline{x}_{ij}/(1 - \overline{x}_{i_\circ j})$ for each $j \in \mathcal{D}$, $i \in \mathcal{F} - \{i_\circ\}$, and $\widehat{y}_i = \overline{y}_i/(1 - \overline{y}_{i_\circ})$, $\widehat{y}_{i\circ} = 0$. Then, it is easy to see that $(\widehat{x}, \widehat{y})$ is feasible. Furthermore, $p(\widehat{x}, \widehat{y}) \leq \frac{1}{1-\overline{y}_{i_\circ}}(p(\overline{x}, \overline{y}) - f_{i_\circ}\overline{y}_{i_\circ}) < \frac{1}{1-\overline{y}_{i_\circ}}(p(\overline{x}, \overline{y}) - \overline{y}_{i_\circ} p(\overline{x}, \overline{y})) = p(\overline{x}, \overline{y})$.

Lemma 3. *For each client $j \in \mathcal{D}$, let $\mathsf{LB}_j = \min_{i \in \mathcal{F}}(c_{ij} + f_i)$, and let $\mathsf{LB} = \max_{j \in \mathcal{D}} \mathsf{LB}_j$. Then $\mathsf{LB} \leq \mathsf{OPT} \leq n\ \mathsf{LB}$, and there is a feasible solution to* P *whose cost is also bounded by $(n\ \mathsf{LB})$.*

Proof. It is clear that LB_j is a lower bound for each client $j \in \mathcal{D}$. For the upper bound consider the solution obtained opening all the facilities that realize the minima of LB_j ($j \in \mathcal{D}$). Then, the solution is feasible and its cost is at most $\sum_j \mathsf{LB}_j \leq (n\ \mathsf{LB})$.

The following result is due to Young ([26]). Suppose that we have a decision procedure A-FEAS, such that given an error $\epsilon' > 0$ and a target $R > 0$, it either finds a feasible solution x such that $p(x) < (1 + \epsilon')R$ or concludes that $\mathsf{OPT} > R$. The running time of the procedure depends on the size of the input and a polynomial on $\frac{1}{\epsilon'}$ of the form $\frac{1}{\epsilon'} q(1/\epsilon')$ (but *not* on R).

Lemma 4. *Suppose that we know a feasible solution x and a lower bound* LB *such that $\mathsf{LB} \leq \mathsf{OPT} \leq p(x) \leq t\ \mathsf{LB}$, for some $t > 0$, then we can find a solution*

[2] The idea of using a variant of the weighted median algorithm was pointed out to us by one of the referees.

$\overline{x}$ such that $p(\overline{x}) \leq (1+\epsilon)\mathsf{OPT}$ by running A-FEAS $O(\log\log t)$ times with $\epsilon' = \frac{1}{2}$ and an additional number of A-FEAS runs whose overall running time is $O(1)$ times the time necessary to run once A-FEAS with $\epsilon' = \epsilon$.

Now we eliminate the dependence of our algorithm on F. Given $\epsilon, R > 0$, we design A-FEAS as follows: first remove the facilities with $f_i > R$, then apply the algorithm of Theorem 2 for $N = \lceil \frac{2\sqrt{(n-1)(m-1)}}{\epsilon} - 1 \rceil$ iterations. If $p(x_N) \leq (1+\epsilon)R$, we return x_N; else we claim that $\mathsf{OPT} > R$. To see that A-FEAS is a valid decision procedure, note that if $\mathsf{OPT} \leq R$, because of Theorem 2 and Lemma 2 (OPT does not change), we have that

$$p(x_N) \leq \epsilon\ R + d(u_N) \leq \epsilon\ R + \mathsf{OPT} \leq (1+\epsilon)R.$$

Finally we apply Lemma 4 using the initial solution given in Lemma 3 to obtain an algorithm that finds a $(1+\epsilon)$-approximate solution in $O(\sqrt{nm}\ (\log\log n + \frac{1}{\epsilon}))$ iterations. The following theorem improves the dependence on ϵ of the algorithms in [25, 10, 7] by a factor of $O(1/\epsilon)$, which had a fastest running time of $O\left(nm^2 \log n(\log\log n + \frac{1}{\epsilon^2})\right)$.

Theorem 3. *There is an algorithm for the linear programming relaxation of UCFLP that given $\epsilon > 0$ finds a feasible solution $x \in \Delta_m^n$ with objective value $p(x) \leq (1+\epsilon)\mathsf{OPT}$ in $O(\sqrt{nm}\ (\log\log n + \frac{1}{\epsilon}))$ iterations each of which can be implemented in $O(mn)$ time.*

We conclude this section by noting that the soft-capacities variant of UCFLP (now facilities have a capacity, but we can open many copies) can be easily handled with our methods as well. This case is interesting because typical subgradient algorithms used in practice have problems when variables are allowed to take values greater than 1.

3 A Generic Paradigm

In this section, we summarize and generalize our arguments of the previous section. For simplicity of the exposition, we describe the process as follows.

1. Modeling.
2. Solving the problem within an additive error of ϵ.
3. Using binary search to obtain a relative error of $(1+\epsilon)$.

We point out that this type of approach has been followed before in the literature (see Introduction), and our contribution consists only of a generalization that includes the work of [21, 19].

Step 1 is the most important and difficult since it has to take into account what the other steps can do. To be specific, consider the following LP written in the form $\min\{c^t x : Ax \geq b : x \in Q_1\}$, where we assume that $c \in \mathbb{R}^n_+$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $Q_1 \subseteq \mathbb{R}^n_+$ is a polyhedron. Then for multipliers $u \in \mathbb{R}^m_+$, the Lagrangian relaxation is given by $L(u) = \min_{x\in Q_1} c^t x + u^t(b - Ax)$. As we

argued in Section 2, solving the original LP is equivalent to solving the problems $\min_{x\in Q_1}\{p(x) := \max_{u\in Q_2}\{c^t x + \langle -Ax, u\rangle + u^t b\}\}$ and $\max_{u\in Q_2}\{d(u) := \min_{x\in Q_1}\{c^t x + \langle -Ax, u\rangle + u^t b\}\}$ (here $Q_2 = \mathbb{R}^m_+$). Since we want to apply the approach of [21, 19] ideally we have to make sure that Q_1 and Q_2 are bounded (e.g., in Section 2, $Q_1 = \Delta^n_m$, $Q_2 = \Delta^m_n$), but sometimes we may need to delay this to Step 2 when we make additional assumptions about OPT. In Step 2 we have to choose norms and the strongly convex functions d_1 and d_2, and then design the feasibility procedure A-FEAS. Thus, under the assumption that $\mathsf{OPT} \le R$ we have to show that the right hand-side of (3) is a polynomial in the input times R divided by $N + 1$. In general, we accomplish this by restricting our primal (or dual) solution using the bound on OPT (in Section 2, we eliminated the facilities i with $f_i > R$; in other problems, we only consider feasible points within a ball -for some appropriate norm- of radius $O(R)$). In addition, we have to argue how to solve the subproblems of Theorem 1 exactly. Step 3 is very concise and self-contained in Lemma 4. The problem at this point is to find a good enough initial feasible solution.

In the next subsections we show three examples on how to apply the paradigm we described above.

3.1 Packing Problems

In this section we consider the packing problems considered in [6] and recast their algorithm using our paradigm. These are of the form $\min_{\{x\in Q\}} \max^m_{i=1} a^t_i x$, with $a_i \in \{0, 1\}^n$, and $Q \subseteq \mathbb{R}^n_+$ a polyhedron (we need also that for each j, there is an i with $(a_i)_j = 1$). Following our paradigm we first reformulate the problem as a Lagrangian relaxation, which (proceeding as in Section 2) takes the simple form $\min_{\{x\in Q\}}\{\max_{u\in\Delta_m} \sum^m_{i=1}(a^t_i x)u_i\} = \min_{\{x\in Q\}}\{\max_{u\in\Delta_m}\langle Ax, u\rangle\}$, where the i^{th} row of A is a^t_i. Q may not be bounded, however under the assumption that $\mathsf{OPT} \le R$ (start of Step 2), we can restrict our attention to the sets $Q(R) = \{x \in Q : x_j \le R \text{ for all } j\}$ (this is the main idea in [6]).

Now we continue with Step 2. In the primal space x we take the Euclidean norm together with $d_1(x) = \frac{1}{2}||x - x_\circ||^2_{①}$, where $x_\circ \in Q(R)$ ($\sigma_1 = 1$). In the dual space u we use the norm $||u||_{②} = \sum^m_{i=1} |u_i|$, together with the entropy distance, $d_2(u) = \sum^m_{i=1} u_i \ln u_i + \ln m$ (here too $\sigma_2 = 1$, see [21]). Then if K is the maximum number of nonzeros in any row, it follows that $||A||_{①,②} = \max_{||x||_{①}\le 1} \max^m_{i=1} |a^t_i x| \le \sqrt{K}$, $D_1 \le 2nR^2$ and $D_2 = \log m$. So that (3) gives an upper bound on the number of iterations of $O(\sqrt{nK \log m}/\epsilon)R$ for an additive error of ϵR.

In Step 3 we consider the initial solution given in [6] with value no more than $\min\{m, K\}\mathsf{LB}$ for a valid lower bound LB on the true optimum. Then using Lemma 4, we obtain the bound of $O(\sqrt{nK \log m}(\frac{1}{\epsilon} + \log\log\min\{m, K\}))$ iterations (exactly as in [6]). The subproblems we need to solve at each iteration are of the same type as in [6]: convex quadratic problems in Q with the cost function separable (i.e., no product terms of the form $x_i x_j$) and simple problems on the simplex that take $O(m)$ time (they correspond with evaluating the potential function in [6]). We point out that the question of how to solve the primal

problems (that have the form $\min_{x\in Q(R)}\{||x-y||^2_{①}+\alpha^t x\}$) is not addressed in [6] in detail. We discuss this in next section.

In some cases we can remove the factor of $\sqrt{K}$ in the number of iterations by choosing a better norm in the primal space x and keeping the dual norm as before. For $x\in\mathbb{R}^n_+$, let $f(x)=\max_i a_i^t x=\max_{u\in\Delta_m}\langle Ax,u\rangle$. In [23] it is shown how to find a diagonal Euclidean norm, $||x||_{\mathcal{D}}$, in $O(n^2m(\ln n+\ln m))$ time, such that $||x||_{\mathcal{D}}\le f(x)$ for any $x\in\mathbb{R}^n_+$ and $||A||_{\mathcal{D},②}\le 2\sqrt{n}$. Let $d_1(x)=\frac{1}{2}||x-x_\circ||^2_{\mathcal{D}}$ (where $x_\circ$ is the projection of 0 into Q using $||\ ||_{\mathcal{D}}$). If we use this norm now, under the assumption that $\mathsf{OPT}\le R$, we can restrict our problem to $Q(R)=\{x\in Q:||x||_{\mathcal{D}}\le R\}$ (since $||x^*||_{\mathcal{D}}\le f(x^*)=\mathsf{OPT}$ for an optimal solution x^*). If we plug in formula (3), we would immediately improve the bound of [6] by a factor of $\sqrt{K}$ (since now $D_1\le 2R^2$). Unfortunately, the subproblems in $Q(R)$ may be too difficult to solve (now the constraints are nonlinear). In one important case this is possible, however, when $Q=\Delta_n$. Now our problem is equivalent to the linear fractional packing problem [23], that is, $\max\{\overline{c}^t x:\overline{a}_i^t x\le\overline{b}_i, x\ge 0\}$ where all the entries are nonnegative. In this case, we can improve the result of [23] by a factor of $O(\log\sqrt{n}/\log\log\sqrt{n})$ by recasting the problem and using our generic paradigm.

Theorem 4. *After computing a diagonal rounding in $O(n^2m(\ln n+\ln m))$ time, the fractional packing problem can be approximated within a relative gap of $(1-\epsilon)$ in $O(\sqrt{n}\ln m\ (\log\log\sqrt{n}+\frac{1}{\epsilon}))$ iterations, each of which requiring $O(n+m)$ time.*

3.2 Scheduling Jobs on Unrelated Parallel Machines

Here we consider the linear programming relaxation of the following scheduling problem (known as $R||C_{\max}$), a special case of packing that we can solve efficiently. There are m machines and n jobs that need to be processed on the machines without preemption. The processing time of job j if processed by machine i is $p_{ij}>0$ $(1\le i\le m,\ 1\le j\le n)$ and the goal is to schedule all the jobs on the machines so as to minimize the makespan (i.e., the last job completion time). Since x_{ij} is 1 when job j is assigned to machine i, and 0 otherwise, it is easy to see (using the same notation as in Section 2) that $\min_{x\in\Delta^n_m}\{\max\{\max_i\sum_j p_{ij}x_{ij},\max_j\sum_i p_{ij}x_{ij}\}\}$ is a linear programming relaxation. Indeed, using [24] and the filtering technique of [16] the value of this relaxation is at most within a factor of 4 of the integer optimum. The Lagrangian relaxation is $\min_{x\in\Delta^n_m}\{\max_{u\in\Delta_{m+n}}\{\sum_i u_i\sum_j p_{ij}x_{ij}+\sum_j u_{m+j}\sum_i p_{ij}x_{ij}\}\}$. As before, in the dual space u we use the norm $||u||_{②}=\sum_{i=1}^{m+n}|u_i|$, together with the entropy distance, $d_2(u)=\sum_{i=1}^{m+n}u_i\ln u_i+\ln(m+n)$, $\sigma_2=1$, $D_2=\log(n+m)$. In the primal space x we choose the norm $||x||_{①}=\frac{1}{\sqrt{m}}(\sum_{i,j}p_{ij}^2x_{ij}^2)^{1/2}$ and $d_1(x)=\frac{1}{2}||x-x_\circ||^2_{①}$ ($x_\circ$ is the projection of 0 into Δ^n_m using the primal norm, $\sigma_1=1$). Then we have that if $f(x)=\max\{\max_i\sum_j p_{ij}x_{ij},\max_j\sum_i p_{ij}x_{ij}\}$, for any $x\in\Delta^n_m$, $||x||_{①}\le f(x)$ and $||A||_{①,②}\le\sqrt{nm}$ (as sought in [23]). Now we use the same idea as before to construct a feasibility procedure. If $\mathsf{OPT}\le R$, using (3) in the restricted set $Q(R)=\{x\in Q:||x||_{①}\le R\}$ (so $D_1\le 2R^2$), we need $O(\sqrt{nm\ln(n+m)}/\epsilon)$ iterations to get a feasible solution

of value at most $R(1+\epsilon)$. Concerning the subproblems, the quadratic programs minimizing over x take again $O(nm)$ time (generalizing Lemma 1), and the problems minimizing over u take $O(n+m)$ time. It is easy to find a feasible solution and a lower bound with $t = n$ satisfying the conditions of Lemma 4 (assign each job to the machine with shortest processing time). The previously known algorithm for this problem is described in [13] and has a running time of $O(\frac{(m+n)\log(n+m)}{\epsilon^2}mn)$.

Theorem 5. *There is an approximation algorithm for the linear programming relaxation of $R||C_{\max}$ that needs $O(\sqrt{nm\ln(n+m)}(\frac{1}{\epsilon}+\log\log n))$ iterations requiring each $O(nm)$ time to find a feasible solution within $(1+\epsilon)$ of optimum.*

3.3 Set Covering Problem

In this section we consider the set covering problem, which for simplicity we think as the special case of UCFLP when $c_{ij} \in \{0, \infty\}$ and only consider the case when n is substantially greater than m. We first eliminate the variables x_{ij} from the linear programming relaxation. Let $N(j) := \{i \in \mathcal{F} : c_{ij} < \infty\}$ be the set of facilities that can be assigned to client $j \in \mathcal{D}$, and let $e^j \in \mathbb{R}^m$ be the incidence vector of $N(j)$, that is, $(e^j)_i = 1$ if $i \in N(j)$ and 0 otherwise. Then the LP we need to solve takes the form: $\min\{\sum_{i\in\mathcal{F}} f_i y_i : (e^j)^t y \geq 1$ for each $j \in \mathcal{D}$, $y_i \leq 1$ for each $i \in \mathcal{F}$, $y \in \mathbb{R}^m_+\}$. Next, if we let $e \in \mathbb{R}^m$ be the vector of all 1's and use the change of variables $y = e - z$, we can apply Lemma 2 and the algorithm of Theorem 4 to obtain the following result, which improves by a factor $1/\epsilon$ the running times in [25, 10, 7, 13] $\left(O\left(m^2 n\log(n)(\log\log n + \frac{1}{\epsilon^2})\right)\right)$.

Theorem 6. *There is an approximation algorithm for the linear programming relaxation of the set covering problem that in $O(m^{1.5}\sqrt{\ln n}(m+n)/\epsilon) + O^*(m^2 n)$ time computes a feasible solution within a factor of $(1+\epsilon)$ of optimum.*[3]

4 Approximating the Subproblems

A drawback of the method of the previous section (i.e., those in [19, 22]) is that the subproblems of Theorem 1 need to be solved exactly for the bound (3) to be valid. However, in many important cases, even though the subproblems are generally much smaller than the original problem, they may be hard to solve exactly. For example, in the special case of multicommodity flow problems, if we choose an Euclidean norm, typically, the subproblems are single-commodity minimum cost flow problems with convex quadratic and separable objective function (see [6] for the special case of the maximum concurrent multicommodity flow problem). Minoux [17] showed that these subproblems can be solved exactly, however the final step of his algorithm can be very costly (in [17], the ellipsoid method

[3] The notation $O^*(.)$ hides logarithmic factors in or .

is used). Thus, to make the technique of Section 3 available to more general problems we investigate how the precision of the solution of the subproblems affects the overall performance of the algorithms. Our results are generalizations of [19, 22, 6]. Under the assumption that the subproblems can be solved up to an additive error of δ in $O(\log \frac{1}{\delta})$ time (times a polynomial on the size of the input), the bound on the number of iterations (3) remains essentially unchanged. But now each iteration depends on $O(\log \frac{1}{\epsilon})$ by choosing $\delta = \delta(\epsilon)$ (see Theorem 7 below), and thus the overall dependence on ϵ of the running time of the algorithms deteriorates by that factor. We can prove the following generalization of Theorem 1 by strengthening the arguments of [6] and providing error bounds for the primal and dual subproblems.

Theorem 7. *Suppose that the problems that need to be solved in Theorem 1 are solved up to an additive error of δ. Then the modified algorithm finds a sequence of points (x_k, u_k) that essentially satisfy (3) (e.g. replace 4 by 5), if we choose δ with $O(\delta) = O(\frac{1}{(N+1)^4} \frac{1}{||A||_{1,2}} (\frac{\sigma_1 \sigma_2}{D_1 D_2})^{1/2})$.*

We omit the proof in this extend abstract. It essentially uses the fact that an approximate minimizer of a strongly convex function is close to the true minimizer.

Using the theorem, if $\epsilon > 0$ is given, we can find N so that the right hand side of (3) is less than ϵ, and then we choose δ using this value of N. In this way, we get an additive error of ϵ and the dependence on ϵ of the number of iterations is $O(1/\epsilon)$. Notice that to run the algorithm we need to specify a procedure to approximately solve the subproblems and to get an estimate of the overall running time we need to estimate how long it takes to do that. Under reasonable assumptions, the input size for each subproblems in the theorem can be bounded by $O(\log(1/\epsilon))$. We will further discuss this issue in Section 4.2.

For the special case of the setting of [17] (see also [6]), the subproblems can be solved in $O(\log(1/\epsilon))$ times a polynomial on the size of the input, then the theorem gives essentially the result of [6] for the maximum concurrent multicommodity flow problem.

4.1 Network Design Problems

Here we consider the survivable network design problem (SNDP). In this problem we are given an undirected graph $G = (V, E)$ with edge costs $f_e \geq 0$ $(e \in E)$. In addition, for each pair $i, j \in V$, $i \neq j$, there is a connectivity requirement $r_{ij} \in \mathbb{Z}_+$. A feasible solution to the problem consists of a subgraph that for each pair $i, j \in V$, $i \neq j$, contains r_{ij} edge-disjoint paths, and the objective is to find a minimum cost feasible solution. This problem and many of its variations have been studied extensively and a linear programming relaxation has proven very useful for the design of approximation algorithms (see [12] and references thereafter). Recently this linear programming relaxation was treated in [9] and the approach was to consider the exponential version of the LP that had the additional difficulty that the variables were constrained to be at most 1.

Here we proceed differently and solve the following multicommodity flow generalization of the problem (using the max flow-min cut theorem). Now $G = (V, E)$ is a directed graph with edge costs, $f_e \geq 0$ $(e \in E)$, and edge capacities all equal to 1. We also have k commodities. Each commodity i has a source-sink pair (s_i, t_i) and a demand $d_i > 0$. Now we are looking for a minimum cost subgraph that can route all the demands. We will use $x_i(e)$ to denote the amount of flow of commodity i that is routed on edge e, and y_e indicates whether edge e is picked. We will use $\mathcal{N}_i \subseteq \mathbb{R}_+^m$ to denote the convex polytope representing the flow constraints for commodity i with the capacity constraints of 1, that is, if $y \in \mathcal{N}_i$, y satisfies the conservation of flow constraints for every node $v \notin \{s_i, t_i\}$ and s_i is a source with demand $\geq d_i$ of commodity i and $y_e \leq 1$ $(e \in E)$. Thus, $(x_i(e))_e \in \mathcal{N}_i$, and $x = (x_i(e))_{i,e} \in \mathcal{N}_1 \times \ldots \times \mathcal{N}_k$ means that x is a feasible multicommodity flow. Thus we would like to solve $\min\{\sum_{e \in E} f_e y_e : x \in \mathcal{N}_1 \times \ldots \times \mathcal{N}_k,\ x_i(e) \leq y_e\ (e \in E,\ i = 1, \ldots, k), y_e \geq 0 (e \in E)\}$. This LP is a slight modification of the one given in [12] (it has constraints of the form $x(e) \leq 1$ and uncapacitated $\mathcal{N}_i$'s, we move the capacity constraints inside $\mathcal{N}_i$) and allows us to get away without being concerned about the hard variable upper bounds of [9] (her LP and ours are equivalent for SNDP). As in Section 2, we choose Euclidean norms in the primal and dual spaces. If $|V| = n$, $|E| = m$ and $F = \max_{e \in E} f_e$, then again $||A||_{①,②} \leq F$ and here $D_1 = O(km)$, $D_2 = O(m)$. The dual subproblems we need to solve are as in Section 2, so we can solve them exactly. Using ideas from [6], we approximately solve the primal subproblems via piece-wise linear functions and use Theorem 7 to derive an algorithm with an additive error. The extension of Lemma 2 is in [9]. An initial solution satisfying the conditions of Lemma 4 is readily available by considering each commodity separately. As in [6] the binary search step is changed to accommodate the approximate solutions of the subproblems (e.g., ϵ' is always taken as a positive or negative power of 2) and for simplicity we choose the accuracy δ as a function of the target final accuracy ϵ and F that works all the time. Below D is the maximum demand. The following improves the dependence on ϵ of the algorithm described in [9], whose running time is $O(m(m + n \min\{n, k\} \log n) \log n/\epsilon^2 + \min\{n, k\} T_n^m)$, where T_n^m is the time needed to compute a single minimum cost flow problem.

Theorem 8. *Given $\epsilon > 0$, there is an algorithm that finds a feasible solution to the SNDP within a factor of $(1 + \epsilon)$ of optimum in $O(m\sqrt{k}(\log \log k + \frac{1}{\epsilon}))$ iterations plus the time to solve k single commodity maximum flow problem with unit capacities (to find an initial solution) of $O(mnk)$ [1]. Each iteration can be implemented in $O^*((\log(1/\epsilon) + \log F + \log D)km)$ shortest path computations, or $O^*((\log(1/\epsilon) + \log F + \log D)km(m + n \log n))$ overall time.*

Furthermore we can provide similar results for several extensions of the problem such as more general single-commodity flow problems per commodity, routing costs and soft-capacities.

4.2 Discussion

Here we address the issue of solving the subproblems for more general cases, in which the structure of the feasible region is not as benign as the case of a

network flow problem. In such cases, for instance, we could always use an interior point method to approximately solve the subproblems. In general, if we keep $Q(R)$ polyhedral, we can always approximately solve our subproblems in $O(\log^2(1/\epsilon))$ times a log on the size of the data (ϵ excluded). The additional log factor comes from the size of the numbers generated by the algorithm which are of size $O(\log(1/\epsilon))$ times a polynomial on the size of the data (notice that this additional log factor did not affect the algorithm of [6] or Theorem 8 because, essentially, there is a strongly polynomial time algorithm to compute shortest paths). Thus, to get an algorithm with dependence on $\epsilon > 0$ of $O((1/\epsilon)\log^2(1/\epsilon))$ we only need to worry about the problems we initially had in Section 3 (i.e., finding bounds on Lagrangian multipliers, polynomial -or optimum dependent- bounds on the norm operator, D_1 and D_2, and finding a good initial solution). From a practical point of view, Theorem 7 suggests that the crude implementation of the algorithms that lets a commercial standard solver (such as Cplex or Mosek) deal with the strongly convex subproblems of the algorithms of [19, 22] may produce very good solutions. Of course, this approach makes sense only if the subproblems are substantially smaller than the original problem.

5 Computational Experiments

In this section we present a preliminary report on computational experiments for the UCFLP using the algorithm of Section 2 and a few of its variants obtained by changing update rules and norms. We primarily compared the number of iterations that each algorithm required. Our instances were generated randomly following [4].

Among the variations of the implementation of the algorithm of Section 2 (e.g, using the entropy distance), the algorithm that used Euclidean norms typically outperformed all the others as well as substantially its theoretical bounds. On the other hand, the algorithms that used the entropy distance performed fairly poorly. This fact seems quite interesting since many algorithms that were implemented using exponential functions (which map exactly to the entropy distance) were reported to have slow convergence in practice.

The algorithm of [4] is a very fine-tuned subgradient algorithm (based on the volume algorithm of [3]), and produces typically only approximate primal solutions. In general, our best algorithm (the one we describe in Section 2) turn out to be very competitive and in some cases superior to the one in [4].

Acknowledgments

This paper benefited greatly from stimulating discussions with Yurii Nesterov. We also want thank David Shmoys for useful conversations and an anonymous referee for pointing out to us reference [15] and that the idea behind the linear time algorithm for the weighted median problem could be applied to computing projections into a simplex.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] M.L. Balinski. Integer programming: methods, uses, computation. *Management Science*, 12(3):253–313, 1965.

[3] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with the subgradient method. Technical report, IBM Watson, 1998.

[4] F. Barahona and F. Chudak. Solving large scale uncapacitated facility location problems. In P. M. Pardalos ed., *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*. Kluwer Acad. Publ, 1999.

[5] D. Bienstock. *Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice*. ISSN-0771 3894. CORE Lecture Series, U. Catholique de Louvain, Belgium, 2001.

[6] D. Bienstock and G. Iyengar. Solving fractional packing problems in $^*(\frac{1}{\epsilon})$ iterations. In *STOC'04*, pages 146–155, 2004.

[7] F. Chudak. A simple approximation scheme for the linear programming relaxation of the uncapacitated facility location problem based on a recent algorithm of Garg and Khandekar. Manuscript, 2003.

[8] F.A. Chudak and D.B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2003.

[9] L. Fleischer. A fast approximation scheme for fractional covering problems with variable upper bounds. In *SODA'04*, pages 1001–1010, 2004.

[10] N. Garg and R. Khandekar. Fast approximation algorithms for fractional steiner forest and related problems. In *FOCS'02*, pages 500–509, 2002.

[11] N. Garg and J. Konemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *FOCS'98*, pages 300–309, 1998.

[12] K. Jain. A factor 2 approximation for the generalized steiner network problem. *Combinatorica*, 21(21):39–60, 2001.

[13] R. Khandekar. *Lagrangian Relaxation Based Algorithms for Convex Programming Problems*. PhD thesis, Indian Institute of Technology Delhi, 2004.

[14] P. Klein and N. Young. On the number of iterations for dantzig-wolfe optimization and packing-covering approximation algorithms. In *IPCO'99*, pages 320–327, 1999.

[15] B. Korte and J. Vygen. *Combinatorial Optimization : Theory and Algorithms*. Springer, 2000.

[16] J.H. Lin and J.S. Vitter. -approximation with minimum packing constraint violation. In *STOC'92*, pages 771–782, 1992.

[17] M. Minoux. A polynomial algorithm for minimum quadratic cost flows. In *European J. Oper. Res. 18*, pages 377–387, 1984.

[18] P. Mirchandani and R. Francis, eds. *Discrete Location Theory*. John Wiley and Sons, Inc., New York, 1990.

[19] Yu. Nesterov. Excessive gap technique in non-smooth convex minimizarion. *SIAM J. Optimization*, 2003. To appear.

[20] Yu. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2003.

[21] Yu. Nesterov. Smooth minimization of non-smooth functions. *Mathematcal Programming*, 2003. To appear.

[22] Yu. Nesterov. Unconstrained convex minimization in relative scale. Technical report, CORE, 2003.

[23] Yu. Nesterov. Rounding of convex sets and efficient gradient methods for linear programming problems. Technical report, CORE, 2004.
[24] D.B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.
[25] N. Young. K-medians, facility location, and the chernoff-wald bound. In *STOC'00*, pages 86–95, 2000.
[26] N. Young. Sequential and parallel algorithms for mixed packing and covering. In *FOCS'01*, pages 538–546, 2001.

On the Approximability of the Minimum Congestion Unsplittable Shortest Path Routing Problem

Andreas Bley

Konrad-Zuse-Zentrum für Informationstechnik,
Takustr. 7, D-14195 Berlin, Germany
bley@zib.de

Abstract. We are given an undirected simple graph $G = (V, E)$ with edge capacities $c_e \in \mathbb{Z}_+$, $e \in E$, and a set $K \subseteq V^2$ of commodities with demand values $d_{(s,t)} \in \mathbb{Z}_+$, $(,) \in K$. An unsplittable shortest path routing (USPR) of the commodities K is a set of flow paths $_{(s,t)}$, $(,) \in K$, such that each $_{(s,t)}$ is the unique shortest $(,)$-path for commodity $(,)$ with respect to a common edge length function $= (_e) \in \mathbb{Z}_+^E$. The minimum congestion unsplittable shortest path routing problem (Min-Con-USPR) is to find an USPR that minimizes the maximum congestion (i.e., the flow to capacity ratio) over all edges. We show that it is $\mathcal{NP}$-hard to approximate Min-Con-USPR within a factor of $\mathcal{O}(|V|^{1-\epsilon})$ for any > 0. We also present a simple approximation algorithm that achieves an approximation guarantee of $\mathcal{O}(|E|)$ in the general case and of 2 in the special case where the underlying graph G is a cycle. Finally, we construct examples where the minimum congestion that can be obtained with an USPR is a factor of $(|V|^2)$ larger than the congestion of an optimal unsplittable flow routing or an optimal shortest multi-path routing, and a factor of $(|V|)$ larger than the congestion of an optimal unsplittable source-invariant routing. This indicates that unsplittable shortest path routing problems are indeed harder than their corresponding unsplittable flow, shortest multi-path, and unsplittable source-invariant routing problems.

The Min-Con-USPR problem is of great practical interest in the planning of telecommunication networks that are based on shortest path routing protocols.

Keywords: Shortest Path Routing, Unsplittable Flow, Computational Complexity, Approximation.

Mathematical Subject Classification (2000): 68Q25, 90C60, 90C27, 05C38, 90B18.

1 Introduction

Most data networks presently employ shortest path routing protocols such as OSPF, IS-IS, or RIP. Shortest path routing offers many advantages in practice,

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 97–110, 2005.

for instance, decentralized and distributed routing algorithms, little administrative overhead, and better scaling properties than classical connection oriented protocols. These advantages however come at a cost. The end-to-end routing paths can be controlled only jointly and indirectly by changing the administrative routing lengths of the network links. Finding lengths that induce a set of globally efficient end-to-end routing paths is a major difficulty in such networks. The shortest path routing paradigm enforces rather complicated and subtle interdependencies among the paths that comprise a valid routing. Additional difficulties are introduced if the communication demands must be sent unsplit (i.e., via uniquely determined shortest paths) through the network – a requirement that is often imposed in order to ensure tractability of end-to-end traffic flows and to prevent package reordering and other unwanted effects of multi-path routing. As links with small routing lengths are preferred by all communication demands, shortest path routing protocols potentially lead to congestion in the network. This may have severe effects on the overall service quality. Network providers therefore put much effort into reducing and balancing the congestion of their network links.

Mathematically, this planning task can be formulated as a minimum congestion unsplittable shortest path routing problem (Min-Con-USPR). The problem input consists of an undirected graph $G = (V, E)$ with capacities $c_e \in \mathbb{Z}_+$ for all $e \in E$, and a set of commodities $K \subseteq V^2$ with demand values $d_{(s,t)} \in \mathbb{Z}_+$, $(s, t) \in K$. A feasible solution is an unsplittable shortest path routing (USPR) of the commodities, i.e., a single flow path $\Phi_{(s,t)}$ for each commodity $(s, t) \in K$ such that each $\Phi_{(s,t)}$ is the unique shortest (s, t)-path with respect to a common edge length function $\lambda = (\lambda_e) \in \mathbb{Z}_+^E$. The objective is to minimize the maximum congestion (i.e., the flow to capacity ratio) over all edges.

In spite of their great practical relevance, unsplittable shortest path routing problems have receive only little attention in the mathematical literature. Ben-Ameur and Gourdin [1] discuss some structural properties of unsplittable shortest path routings. They also devise several (integer) linear programming models to find lengths that induce a prescribed set of shortest paths. Bley [2] shows that the problem of finding small integer routing weights that uniquely induce a prescribed set of shortest paths is computationally hard. Algorithms based on local search techniques, Lagrangian relaxation, and integer programming methods as well as computational results for real-world network design and congestion minimization problems with unsplittable shortest path routing are presented in [3, 4, 5, 6, 7, 8]. Results concerning the approximability of these problems have not been published (to our knowledge).

In this paper, we prove that it is $\mathcal{NP}$-hard to approximate Min-Con-USPR within a factor of $\mathcal{O}(|V|^{1-\epsilon})$ for any $\epsilon > 0$. In the special case where all demand values and edge capacities are one, Min-Con-USPR remains inapproximable within $2 - \epsilon$. Furthermore, we present two simple $\mathcal{O}(|E|)$-approximation algorithms for Min-Con-USPR. In the special case where the underlying graph is a cycle, both algorithms achieve an approximation guarantee of 2. In the last section, we construct examples where the gap between the minimum congestion

value for USPR and the minimum congestion for unsplittable flow routing is as bad as $\Omega(|V|^2)$. We also show that USPR may be $\Omega(|V|^2)$ worse than shortest multi-path and multicommodity flow routing and $\Omega(|V|)$ worse than unsplittable source-invariant routing. This gives theoretical evidence for the practical experience that routing planning is harder for USPR than for shortest multi-path routing, unsplittable source-invariant routing, and unsplittable flow routing.

2 Notation and Preliminaries

Let $G = (V, E)$ be an undirected graph with edge capacities $c_e \in \mathbb{Z}_+$, $e \in E$ and let $K \subseteq V^2$ be a set of (undirected) commodities with demand values $d_{(s,t)} \in \mathbb{Z}_+$, $(s,t) \in K$. An edge length function $\lambda = (\lambda_e) \in \mathbb{Z}_+^E$ defines an *unsplittable shortest path routing (USPR)* for the commodity set K if the shortest (s,t)-path $\Phi_{(s,t)}(\lambda)$ with respect to λ is *uniquely determined* for each commodity $(s,t) \in K$. The demand of each commodity is routed unsplit along the respective shortest path. For edge lengths λ that define such an USPR, the total flow through an edge $e \in E$ is

$$f_e(\lambda) := \sum_{(s,t)\in K:\, e\in\Phi_{(s,t)}(\lambda)} d_{(s,t)} \ . \tag{1}$$

The task in the Minimum Congestion Unsplittable Shortest Path Routing problem is to find edge lengths λ that define an USPR for the given commodity set K. The objective is to minimize the maximum congestion (i.e., the ratio $f_e(\lambda)/c_e$) over all edges. Formally, this problem is defined as follows:

Problem:	Min-Con-USPR
Instance:	Graph $G = (V, E)$ with edge capacities $c_e \in \mathbb{Z}_+$, $e \in E$, and commodity set $K \subseteq V \times V$ with demands $d_{(s,t)} \in \mathbb{Z}_+$, $(s,t) \in K$.
Solution:	Edge lengths $\lambda_e \in \mathbb{Z}_+$, $e \in E$, such that the shortest (s,t)-path is uniquely determined for each commodity $(s,t) \in K$.
Objective:	$\min \max_{e\in E} f_e(\lambda)/c_e$, where $f_e(\lambda)$ is as defined in (1).

We may assume without loss of generality that the underlying graph G is connected. Clearly, no commodity $(s,t) \in K$ with s and t in different components can be routed and all nodes and edges in a commodity's routing path must belong to the same component of G. Furthermore, we may assume that G is simple and that there are no parallel commodities. Loops cannot be contained in any uniquely determined shortest path. If two parallel edges were contained in two commodities' routing paths, these paths would not be unique shortest paths. Parallel commodities between two nodes s and t would have to use the same flow path in any unsplittable shortest path routing and therefore can be aggregated into one commodity.

Observe that, for any bijection $\mathrm{idx} : E \leftrightarrow \{1, \ldots, |E|\}$, the edge length function $\lambda_e := 2^{\mathrm{idx}(e)}$ induces unique shortest paths between all node pairs. Hence,

there exists an USPR for any given commodity set K, provided that the underlying graph G is connected.

An assumption that is commonly made for unsplittable flow problems is that the maximum demand value does not exceed the minimum capacity. Typically, unsplittable flow problems are easier to approximate with this additional assumption than in the general case, c.f. [9, 10, 11]. For unsplittable shortest path routing problems this is not the case.

Proposition 1. *For any instance* $\mathrm{I} = (G, c, K, d)$ *of* Min-Con-USPR *with* $d_{\max} > c_{\min}$*, there exists an equivalent instance* $\mathrm{I}' = (G', c', K', d')$ *with* $d'_{\max} \leq c'_{\min}$ *(i.e., any solution for* I *with objective value* L *can be transformed into a solution for* I′ *with objective value* L*, and vice versa).*

Proof. Suppose we are given a Min-Con-USPR instance (G, c, K, d) with $c_{\min} < d_{\max}$. Let $r := d_{\max}/c_{\min}$ and $q := \lceil \sqrt{r} \rceil$. For each node $v \in V$, we introduce q additional nodes v^j and edges vv^j, $j = 1, \ldots, q$. The capacities c' are set $c'_e := c_e$ for all $e \in E$ and $c'_{vv^j} := \sum_{e \in E} c_e$ for all $j = 1, \ldots, q$. Each commodity $(s, t) \in K$ is replaced by q^2 new commodities (s^i, t^j) with demand values $d'_{(s^i,t^j)} \in \{\lfloor d_{(s,t)}/q^2 \rfloor, \lceil d_{(s,t)}/q^2 \rceil\}$ such that $\sum_{i,j=1}^{q} d'_{(s^i,t^j)} = d_{(s,t)}$. Clearly, $d'_{\max} \leq c'_{\min}$ holds.

Because each node v^j, $j = 1, \ldots, q$ has only one neighbor, namely v, all q^2 commodities (s^i, t^j), $i, j = 1, \ldots, q$, are routed via the same (s, t)-subpath in any unsplittable shortest path routing. An unsplittable shortest path routing in G therefore corresponds to an unsplittable shortest path routing in G', and vice versa. As the corresponding routings induce the same flows on the edges of G, the maximum congestion values are equal for both routings. □

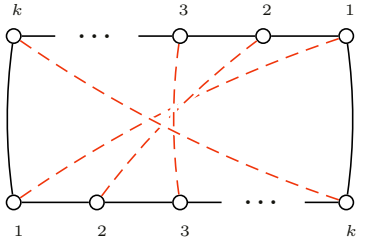
Note that this transformation is not polynomial in the strong sense. The size of the graph grows by a factor of $\Theta(d_{\max}/c_{\min})$.

If the underlying graph G is a forest, then all edge length vectors define the same USPR. In this case, any edge length vector λ is an optimal solution for Min-Con-USPR. The simplest non-trivial case is when G is a cycle. Yet, already in this case Min-Con-USPR is (weakly) $\mathcal{NP}$-hard.

Theorem 2. Min-Con-USPR *is* $\mathcal{NP}$*-hard even if the underlying graph is a cycle.*

Proof. We construct a polynomial reduction from Partition to the problem of solving Min-Con-USPR to optimality. Given a Partition instance consisting of the items $i \in \{1, \ldots, k\}$ with sizes $d_i \in \mathbb{Z}_+$, the instance of Min-Con-USPR is built as shown in Figure 1.

Since the graph is a cycle and all commodity pairs "cross", each unsplittable flow routing of the commodities can be realized as an unsplittable shortest path routing [1]. Therefore, any feasible partition of the items corresponds to an undirected unsplittable shortest path routing of the commodities such that the flows do not exceed the edge capacities, and vise versa. The commodities routed across $s_1 t_k$ form one set of the partition, those routed across $s_k t_1$ the other set. □



For each item $i \in \{1, \ldots, k\}$, we introduce two nodes $\quad_i$, $\quad_i$ and a commodity $(\quad_i, \quad_i)$ with a demand value $d_{(s_i,t_i)} = d_i$. The edge set E consists of the edges $\quad_i \quad_{i+1}$ and $\quad_i, \quad_{i+1}$ for all $i = 1, \ldots, k-1$, as well as $\quad_1 \quad_k$ and $\quad_k \quad_1$. The edge capacities are $c_{s_1 t_k} = c_{s_k t_1} = 1/2 \sum_{i=1}^{k} d_i$, and $c_{s_i s_{i+1}} = c_{t_i t_{i+1}} = \sum_{i=1}^{k} d_i$ for all $i = 1, \ldots, k-1$.

Fig. 1. Reduction from Partition to Min-Con-USPR: Solid lines are edges, dashed lines are commodities

3 Inapproximability Results

In this section, we show that Min-Con-USPR is not approximable within a factor in $\mathcal{O}(|V|^{1-\epsilon})$, unless $\mathcal{P} = \mathcal{NP}$. As a first step, we show that there is no constant factor approximation.

Lemma 3. *Let $\alpha \in \mathbb{Z}_+$ be an arbitrary number. It is $\mathcal{NP}$-hard to approximate* Min-Con-USPR *within a factor less than $\alpha + 1$.*

Proof. We construct a reduction from the $\mathcal{NP}$-complete decision problem Fully Disjoint Paths to Min-Con-USPR. Fully Disjoint Paths is a restricted variant of the classical Disjoint Paths problem [12]. Given an undirected graph $H = (W, F)$ and a set of node pairs (s_i, t_i), $i = 1, \ldots, k$, the task is to find (s_i, t_i)-paths P_i in H that are not only internally disjoint but share no nodes at all (including the paths' first and last nodes), i.e., $\{v \in W \mid v \in P_i \text{ and } v \in P_j\} = \emptyset$ for all $i, j = 1, \ldots, k$ with $i \neq j$. It is easy to verify that the Disjoint Paths problem remains $\mathcal{NP}$-complete with this stronger notion of disjointness.

Suppose we are given a Fully Disjoint Paths instance consisting of the graph $H = (W, F)$ and the node pairs (s_i, t_i), $i = 1, \ldots, k$. W.l.o.g., we may assume that there is an (s_i, t_i)-path in H for each $i = 1, \ldots, k$ and that $\{s_i, t_i\} \cap \{s_j, t_j\} = \emptyset$ for all $i \neq j$.

We construct a Min-Con-USPR instance (G, c, K, d) as follows. The graph $G = (V, E)$ contains all nodes and edges of H. Furthermore, G contains one extra node r and $2k\alpha$ additional nodes nodes u_i^l and v_i^l with $l = 1, \ldots, \alpha$ and $i = 1, \ldots, k$, i.e.,

$$V := W \cup \{r\} \cup \{u_i^l,\, v_i^l \mid i = 1, \ldots, k\,,\ l = 1, \ldots, \alpha\}.$$

For each $i = 1, \ldots, k$, we add $2\alpha + 1$ new edges, which comprise the edge set

$$\begin{aligned} E_\alpha := &\{r v_i^1,\, v_i^\alpha t_i,\, u_i^\alpha s_i \mid i = 1, \ldots, k\} \\ &\cup \{u_i^l u_i^{l+1},\, v_i^l v_i^{l+1} \mid i = 1, \ldots, k\,,\ l = 1, \ldots, \alpha - 1\}. \end{aligned}$$

Additionally, we introduce α^2 edges for each pair $i, j = 1, \ldots, k$ with $i \neq j$. These edges form the set

$$E_1 := \{u_i^l v_j^m \mid i, j = 1, \ldots, k\,,\ i \neq j\,,\ l, m = 1, \ldots, \alpha\}.$$

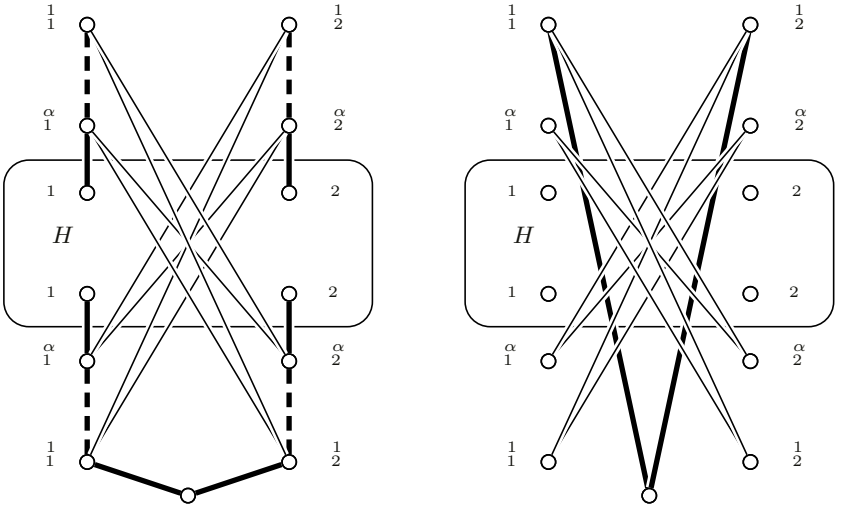
We let $E := F \cup E_\alpha \cup E_1$. The edge capacities are defined as

$$c_e := \begin{cases} 1 & \text{, if } e \in E_1, \text{ and} \\ \alpha & \text{, otherwise.} \end{cases}$$

The commodity set K contains two types of commodities. For each $i = 1, \ldots, k$, there is a commodity (u_i^1, r) with demand value $d_{(u_i^1, r)} = \alpha$. For each pair $i, j = 1, \ldots, k$ with $i \neq j$ and each pair $l, m = 1, \ldots, \alpha$, there is a commodity (u_i^l, v_j^m) with $d_{(u_i^l, v_j^m)} = 1$.

Figure 2 illustrates the constructed Min-Con-USPR instance for the case where $k = 2$. It is obvious that this transformation is polynomial in the encoding size of the given Fully Disjoint Paths instance and α. Furthermore, any edge length function $\lambda \in \mathbb{Z}_+^E$ that induces unique shortest paths for the commodities in K defines a feasible solution for the constructed Min-Con-USPR instance.

In the first part of the proof, we show that there exists an unsplittable shortest path routing of the commodities K that does not exceed the edge capacities c if the given Fully Disjoint Paths instance has a feasible solution. Assume there exist fully disjoint (s_i, t_i)-paths P_i in H, $i = 1, \ldots, k$. Let idx be an arbitrary bijection idx : $F \cup E_\alpha \leftrightarrow \{1, \ldots, |F \cup E_\alpha|\}$ and consider the edge length function λ' on the subgraph $G_P := (V, F \cup E_\alpha)$ defined as



(a)— Graph G with indicated edge capacities: Edges with capacity are bold, edges with capacity 1 are thin

(b)— Commodities K with indicated demand values: commodities with demand are bold, commodities with demand 1 are thin

Fig. 2. Constructed Min-Con-USPR instance

$$\lambda'_e := \begin{cases} 2 & \text{, if } e \in P_i \text{ or } e \in E_\alpha \text{ , and} \\ 2 \cdot |F| + 2^{\mathrm{idx}(e)} & \text{, otherwise.} \end{cases}$$

Since G_P is connected, the distance $\mathrm{dist}(u, v)$ in G_P with respect to λ' is well defined for each node pair $u, v \in V$. We define the edge lengths λ on the entire graph G as

$$\lambda_e := \begin{cases} \lambda'_e & \text{, if } e \in F \cup E_\alpha, \text{ and} \\ \mathrm{dist}(u_i^l, v_j^m) - 1 & \text{, otherwise (i.e., } e = u_i^l v_j^m \in E_1). \end{cases}$$

It is easy to verify that *all* shortest paths in G are unique with respect to λ. In particular, the shortest (u_i^1, r)-path is the path $(u_i^1, \ldots, u_i^\alpha, s_i) + P_i + (t_i, v_i^\alpha, \ldots v_i^1, r)$ for each $i = 1, \ldots, k$, and the shortest (u_i^l, v_j^m)-path is the edge $u_i^l v_j^m$ for each $i, j = 1, \ldots, k$ with $i \neq j$ and $l, m = 1, \ldots, \alpha$. Figure 3 illustrates this routing. Clearly, the induced edge flows do not exceed the given edge capacities c.

In the second part of the proof, we show that the flows of any unsplittable shortest path routing exceed at least one edge capacity by a factor of at least $\alpha + 1$ if the Fully Disjoint Paths instance is not solvable. So, suppose there is no set of fully disjoint (s_i, t_i)-paths in H and let $\lambda \in \mathbb{Z}_+^E$ be an arbitrary edge length function that defines unique shortest (u, v)-paths $\Phi_{(u,v)}(\lambda)$ for all $(u, v) \in K$.

First, assume that some edge $u_i^l v_j^m$ is contained in the shortest path $\Phi_{(u_{i'}^1, r)}(\lambda)$ for some commodity $(u_{i'}^1, r)$. Since $\Phi_{(u_{i'}^1, r)}(\lambda)$ is the unique shortest $(u_{i'}^1, r)$-path, its edge (u_i^l, v_j^m) is also the unique shortest (u_i^l, v_j^m)-path. The total flow across

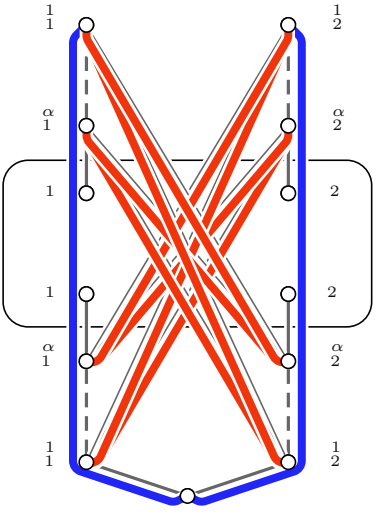


Fig. 3. USPR in G if fully disjoint ($_i$, $_i$)-paths exist in H

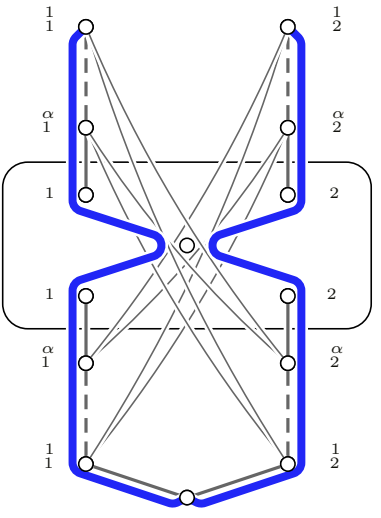


Fig. 4. No USPR. If the shortest $(\begin{smallmatrix}1\\1\end{smallmatrix}, \begin{smallmatrix}\alpha\\1\end{smallmatrix})$- and $(\begin{smallmatrix}1\\2\end{smallmatrix}, \begin{smallmatrix}\alpha\\2\end{smallmatrix})$-paths intersect in some internal node , they are not uniquely determined

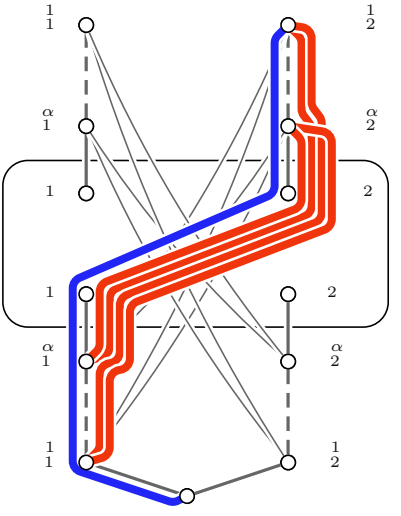


Fig. 5. If $\begin{smallmatrix}1\\\end{smallmatrix} \in {}_{(u_2^1,r)}(\ )$, then the shortest path property forces all commodities $(\begin{smallmatrix}l\\2\end{smallmatrix}, \begin{smallmatrix}m\\1\end{smallmatrix})$ to follow their respective subpath of ${}_{(u_2^1,r)}(\ )$

this edge therefore is at least $d_{(u_{i'}^1,r)} + d_{(u_i^l,v_j^m)} = \alpha + 1$, while its capacity is $c_{u_i^l v_j^m} = 1$.

In the following, we thus may assume that all commodities (u_i^1, r), $i \in \{1,\dots,k\}$, are routed within the subgraph $G_\alpha = (V, F \cup E_\alpha)$. Now, suppose we had $t_i \in \Phi_{(u_i^1,r)}(\lambda)$ for all $i = 1,\dots,k$. Then at least two of these paths, say $\Phi_{(u_1^1,r)}(\lambda)$ and $\Phi_{(u_2^1,r)}(\lambda)$, would have to intersect in some internal node $w \in W$, as illustrated in Figure 4. Otherwise, there would exist fully disjoint (s_i, t_i)-paths in H. However, these two paths cannot be uniquely determined shortest paths, as they contain different subpaths between w and r.

Consequently, there must be some $i = 1,\dots,k$ such that $t_i \notin \Phi_{(u_i^1,r)}(\lambda)$. Since $\Phi_{(u_i^1,r)}(\lambda)$ is completely contained in G_α, there is some $j \neq i$ such that $t_j \in \Phi_{(u_i^1,r)}(\lambda)$. Furthermore, all nodes u_i^l and v_j^m with $l, m \in \{1,\dots,\alpha\}$ are contained in $\Phi_{(u_i^1,r)}(\lambda)$. Hence, all α^2 commodities (u_i^l, v_j^m) are routed along their respective subpath of $\Phi_{(u_i^1,r)}(\lambda)$, see Figure 5. The total flow across the edge $t_j u_j^\alpha$ therefore is at least $\alpha + \alpha^2 \cdot 1$, while its capacity is only α.

Together, the two parts of the proof imply that it is $\mathcal{NP}$-hard to approximate Min-Con-USPR within a factor less than $\alpha + 1$. □

If we choose $\alpha = 1$ in the above construction, all capacities and demand values of the Min-Con-USPR instance are equal to 1. This yields the following theorem.

Theorem 4. *For any $\epsilon > 0$, it is $\mathcal{NP}$-hard to approximate* Min-Con-USPR *within a factor of $2 - \epsilon$, even if all demand values and capacities are equal to one.*

For the general case, we obtain a stronger non-constant inapproximability bound by choosing α depending on the size of the given Fully Disjoint Paths instance.

Theorem 5. *For any $\epsilon > 0$, it is $\mathcal{NP}$-hard to approximate* Min-Con-USPR *within a factor of $\mathcal{O}(|V|^{1-\epsilon})$.*

Proof. It is not difficult to verify that the encoding size of the constructed Min-Con-USPR instance is $\mathcal{O}\big(\alpha^2 \log \alpha(|W| + |F|)\big)$. With $\alpha = \alpha(H) := |W|^q$, the presented construction thus remains polynomial in $|W| + |F|$ for any fixed $q \in \mathbb{Z}_+$. Because $|V| \in \Omega(\alpha)$, there exists some $q_\epsilon \in \mathbb{Z}_+$ for each $\epsilon > 0$, such that $\alpha \notin \mathcal{O}(|V|^{1-\epsilon})$ for $\alpha := |W|^{q_\epsilon}$. With Lemma 3, this implies the claim. □

Analogously, it follows that approximating Min-Con-USPR within a factor of $\mathcal{O}(|E|^{1/2-\epsilon})$ or $\mathcal{O}(\langle \mathrm{I} \rangle^{1/2-\epsilon})$ is $\mathcal{NP}$-hard for any $\epsilon > 0$, where $\langle \mathrm{I} \rangle$ is the encoding size of the Min-Con-USPR instance.

By adding α many new nodes r^j, $j = 1, \ldots, \alpha$, and replacing each commodity (u_i^1, r) of demand $d_{(u_i^1, r)} = \alpha$ by α many commodities (u_i^1, r^j) with $d_{(u_i^1, r^j)} = 1$, we may transform the Min-Con-USPR instance constructed in the proof of Lemma 3 into an instance that satisfies $d_{\max} \leq c_{\min}$. For the class of Min-Con-USPR instances we construct in the proof of Lemma 3, this transformation is strongly polynomial. Therefore, the inapproximability results of Lemma 3 and Theorem 5 also hold for the case where $d_{\max} \leq c_{\min}$.

4 An $\mathcal{O}(|E|)$-Approximation Algorithm

By Theorem 2, Min-Con-USPR is $\mathcal{NP}$-hard to approximate within a factor less than $\mathcal{O}(|V|)$. However, it is easy to compute $\mathcal{O}(|E|)$-approximate solutions.

For each commodity $(s,t) \in K$, let $\mathcal{P}_{(s,t)}$ denote the set of all (s,t)-paths in G. Furthermore, let $\mathcal{P} := \bigcup_{(s,t)\in K} \mathcal{P}_{(s,t)}$. In a fractional multicommodity flow (MCF) routing, the demand of each commodity (s,t) may be distributed arbitrarily among the paths $\mathcal{P}_{(s,t)}$. Such a routing can be expressed as an assignment $x : \mathcal{P} \to [0,1]$, where each x_P denotes the fraction of the demand $d_{(s,t)}$ that is sent along $P \in \mathcal{P}_{(s,t)}$. The problem of finding an MCF routing of minimal congestion can be formulated as follows:

$$\begin{aligned} \min\ & L && && \text{(MCF-LP)} \\ \sum_{P\in\mathcal{P}_{(s,t)}} & x_P = 1 && \forall\,(s,t) \in K \\ \sum_{(s,t)\in K} \sum_{P\in\mathcal{P}_{(s,t)}:e\in P} & d_{(s,t)} \cdot x_P \leq L \cdot c_e && \forall\, e \in E && (2) \\ & L \geq 1 \\ & 0 \leq x_P \leq 1 && \forall\,(s,t) \in K,\ P \in \mathcal{P}_{(s,t)} \end{aligned}$$

Let (L^*, x^*) be an optimal solution of (MCF-LP). Clearly, L^* is a lower bound for the minimum congestion with an unsplittable shortest path routing. By $f_e(x^*) :=$

$\sum_{(s,t)\in K} \sum_{P\in\mathcal{P}_{(s,t)}:e\in P} d_{(s,t)} x^*_P$ we denote the total flow across an edge $e \in E$ in the corresponding multicommodity flow routing.

The idea of the RemoveSmallLinks algorithm is to construct a spanning tree of G by "removing" the least utilized edges. All commodities are routed just within the remaining tree.

RemoveSmallLinks
1. Compute optimal solution (L^*, x^*) of (MCF-LP).
2. Let $F = E$.
3. For all edges $e \in F$ in order of increasing $f_e(x^*)$ do
4 If $F - e$ spans G, then remove e from F
5. Return USPR defined by $\lambda_e := \begin{cases} 1 & \text{, if } e \in F, \\ \lvert F\rvert + 1 & \text{, else.} \end{cases}$

Theorem 6. *RemoveSmallLinks is a $\mathcal{O}(|E|)$-approximation algorithm for* Min-Con-USPR*.*

Proof. Obviously, λ defines an USPR. Let $f_e(x^*)$ and $f_e(\lambda)$ denote the edge flows induced by the optimal solution (L^*, x^*) of (MCF-LP) and by the edge lengths λ returned by RemoveSmallLinks, respectively. For all edges $e \notin F$, we clearly have $f_e(\lambda) = 0$. An edge $e \in F$ receives additional traffic only from those edges that were considered for removal before e. Hence, we have

$$f_e(\lambda) \le \sum_{e': f_{e'}(x^*) \le f_e(x^*)} f_{e'}(x^*) \le |E| \cdot f_e(x^*) \quad \text{for any } e \in F.$$

Consequently, $L^* \le \max_{e\in E} f_e(\lambda)/c_e \le \max_{e\in F} |E| \cdot f_e(x^*)/c_e = |E| \cdot L^*$. □

Let $\pi^*_e \in \mathbb{R}_+$, $e \in E$, be the optimal dual variables corresponding to the constraints (2) in (MCF-LP). It follows from LP duality that all paths P with $x^*_P > 0$ are shortest paths between their respective terminal nodes with respect to the edge lengths π^*_e. However, these paths are not necessarily *uniquely determined* shortest paths and not all shortest paths have a positive flow. The idea behind our second algorithm PenalizeSmallLinks is to start with the optimal dual variables π^*_e and perturb these lengths until the shortest path is unique for each commodity.

PenalizeSmallLinks
1. Compute optimal solution (L^*, x^*) of (MCF-LP). Let π^*_e be the optimal dual variables for (2).
2. Find *integer* weights $\lambda'_e \ge 1$ that induce the same shortest paths as π^*_e. (i.e. all paths P with $x^*_P > 0$).
3. Number the edges $idx : E \to \{1, \ldots, \lvert E\rvert\}$ in order of decreasing $f_e(x^*)$.
4. Return weights $\lambda_e := 2^{\lvert E\rvert+1} \cdot \lambda'_e + 2^{idx(e)}$.

Theorem 7. *PenalizeSmallLinks is a $\mathcal{O}(|E|)$-approximation algorithm for* Min-Con-USPR*.*

It is easy to see that there always exist integer edge lengths that induce exactly the same shortest paths like some given fractional weights (whether or not uniqueness is an issue). They can be computed in polynomial time with linear programming based scaling and rounding methods, as shown in [1]. It is straightforward to verify that the weights λ_e returned by Algorithm PenalizeSmallLinks define an USPR with congestion at most $|E|$ times the congestion of the optimum MCF routing. In the next section, we present examples where the ratio of $\Omega(|E|)$ between the optimal USPR and MCF routings is attained.

If the underlying graph is a cycle, both algorithms need to remove or penalize only one edge until the remaining graph is a path itself or all shortest paths are unique.

Corollary 8. *Both algorithms RemoveSmallLinks and PenalizeSmallLinks achieve a 2-approximation guarantee for* Min-Con-USPR *on a cycle.*

This guarantee extends straightforward to the case where all blocks of the underlying graph are cycles.

5 Relation to Other Routing Problems

The unsplittable shortest path routing model is very restrictive and inherits structural properties of several other routing models. In this section, we compare unsplittable shortest path routing to four closely related but less restrictive routing models. We show that the minimal congestion that can be obtained with unsplittable shortest path routing for a given commodity set may exceed the congestion achievable with the other routing models by an arbitrarily large factor.

The most flexible routing model is (fractional) multicommodity flow routing. With this routing model, the demand of each commodity may be distributed arbitrarily and independent of the other commodities onto several flow paths and thus admits the best possible use of the available capacities. In order to implement a MCF routing in practice, the network must admit the configuration of arbitrary end-to-end routing paths and flow distributions for each commodity individually, which introduces many practical difficulties and complicates the network management. Therefore, many telecommunication network protocols are based on routing models that are less capacity efficient but easier to implement in practice.

With shortest multi-path routing, the traffic that is sent from a node s to a node t is distributed equally to all neighbors of s that are contained in any shortest (s,t)-path with respect to the administrative edge lengths λ. This routing model (adequately) describes so-called equal cost multi-path traffic splitting policies in shortest path routing protocols. Fortz and Thorup [13] show that the minimum congestion shortest multi-path routing cannot be approximated within a factor less than 3/2.

The unsplittable flow routing model requires that each commodity is sent unsplit via a single path through the network. In contrast to the USPR model, the commodities' flow paths may be chosen independent of each other. Kollipoulos and Stein [10] prove that it is $\mathcal{NP}$-hard to approximate the minimum congestion unsplittable flow routing within a factor of $2 - \epsilon$, for any $\epsilon > 0$.

The third routing model that is closely related to USPR is unsplittable source-invariant routing. With this routing model, each commodity is routed on a single flow path. All flow paths with the same destination must form a tree directed towards this destination. Once two flows meet on their way to a common destination, they cannot split anymore. This model describes the routing possibilities of packet networks with independently configurable store-and-forward routers. Lorenz et al. [14] show that finding a minimum congestion unsplittable source-invariant routing is $\mathcal{NP}$-hard. They also show that the minimum congestion may be factor $\Omega(|V|)$ higher for unsplittable source-invariant routing than for unsplittable flow routing.

Let $G = (V, E)$ be a graph with edge capacities $c_e \in \mathbb{Z}_+$ for all $e \in E$, and $K \subseteq V^2$ be a set of commodities with demands $d_{(s,t)} \in \mathbb{Z}_+$ for all $(s, t) \in K$. We denote the optimal solution value of the Min-Con-USPR problem by L^{USPR}. With L^{MCF}, L^{UFP}, L^{SMPR}, and L^{USIR} we refer to the minimal congestion values that can be obtained with fractional multicommodity flow routing, unsplittable flow routing, shortest multi-path routing, and source-invariant routing on the same instance, respectively.

It is obvious that $L^{USPR} \geq L^{UFP} \geq L^{MCF}$, $L^{USPR} \geq L^{SMPR} \geq L^{MCF}$, and $L^{USPR} \geq L^{USIR} \geq L^{MCF}$, since every unsplittable shortest path routing is also a valid shortest multi-path routing, a valid unsplittable flow routing, and a valid unsplittable source-invariant routing of the given commodities. Instances where the gap between unsplittable shortest path routing and the other routing models is large can be constructed straightforward.

Proposition 9. *There is a family of instances with*

(i) $L^{USPR} \geq \Omega(|V|^2) \cdot L^{SMPR}$,
(ii) $L^{USPR} \geq \Omega(|V|^2) \cdot L^{UFP}$, *and*
(iii) $L^{USPR} \geq \Omega(|V|^2) \cdot L^{MCF}$.

Proof. For each $\alpha \in \mathbb{Z}_+$, consider the graph $G = (V, E)$ illustrated in Figure 6. It consists of the nodes $V := \{s, t\} \cup \{s_i, t_i, u_i, v_i \mid i = 1, \ldots, \alpha\}$ and the edges $E := E_\alpha \cup E_1$, where $E_\alpha := \{(s_i, s), (s, u_i), (v_i, t), (t, t_i) \mid i = 1, \ldots, \alpha\}$ and $E_1 := \{(u_i, v_j) \mid i, j = 1, \ldots, \alpha\}$. The edge capacities are $c_e = 1$ for all $e \in E_1$ and $c_e = \alpha$ for all $e \in E_\alpha$. In this graph, consider the commodities $K := \{(u_i, v_j) \mid i, j = 1, \ldots, \alpha\}$ with demands $d_{(u_i,v_j)} = 1$ for all $(u_i, v_j) \in K$.

The congestion of any unsplittable shortest path routing is $L^{USPR} = \alpha^2$, since all commodities' routing paths must follow the same subpath between the nodes s and t in an unsplittable shortest path routing, and therefore share some edge $u_i v_j$ of capacity 1.

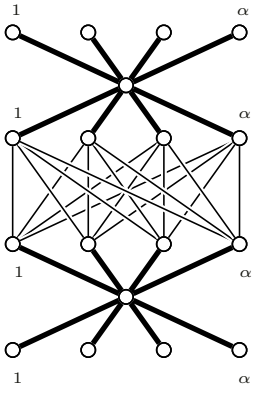


Fig. 6. Instance with $L^{USPR} = {}^{2}$ and $L^{UFP} = L^{SMPR} = L^{MCF} = 1$. Edges with capacity are bold, edges with capacity 1 are thin. For each node pair $_{i\ j}$ there is a commodity with demand 1

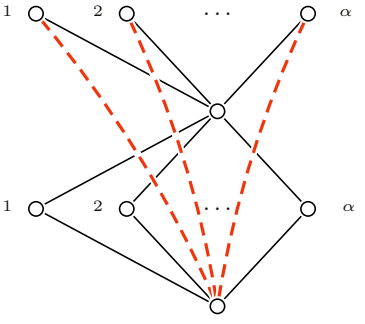


Fig. 7. Instance with $L^{USPR} \geq (|V|)\, L^{USIR}$: The solid lines are edges, the dashed lines are the commodities

On the other hand, the congestion is 1 for an optimal shortest multi-path routing (where all edge lengths are chosen equal), as well as for an optimal unsplittable flow routing or an optimal multicommodity flow routing.

□

Proposition 10. *There is a family of instances with* $L^{USPR} \geq \Omega(|V|) \cdot L^{USIR}$.

Proof. Let $\alpha \in \mathbb{Z}_+$ and consider the graph illustrated in Figure 7. It contains the nodes $V := \{s,t\} \cup \{t_i, v_i \mid i = 1,\ldots,\alpha\}$ and the edges $E := \{(s,v_i), (v_i,t), (t,t_i) \mid i = 1,\ldots,\alpha\}$. All edges have capacity 1. In this network, consider the commodities $K := \{(s,t_i) \mid i = 1,\ldots,\alpha\}$ with demands $d_{(s,t_i)} = 1$. In any unsplittable shortest path routing, all commodities are routed via the same subpath between s and t. The minimal congestion value for USPR therefore is $L^{USPR} = \alpha$. With source-invariant routing, the commodities may be routed via different s,t-subpaths, as they have different destinations. The optimal congestion for this routing model therefore is 1. □

6 Concluding Remarks

Our inapproximability results carry over straightforward to the directed case, as well as the proven gaps between the different routing paradigms. The special case where the underlying digraph is a bidirected ring can be approximated within a factor of 3.

References

[1] Ben-Ameur, W., Gourdin, E.: Internet routing and related topology issues. SIAM Journal on Discrete Mathematics **17** (2003) 18–49
[2] Bley, A.: Inapproximability results for the inverse shortest paths problem with integer lengths and unique shortest paths. Technical Report ZR-05-04, Konrad-Zuse-Zentrum für Informationstechnik Berlin (2005)
[3] Bley, A., Grötschel, M., Wessäly, R.: Design of broadband virtual private networks: Model and heuristics for the B-WiN. In Dean, N., Hsu, D., Ravi, R., eds.: Robust Communication Networks: Interconnection and Survivability. Volume 53 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science., American Mathematical Society (1998) 1–16
[4] Bley, A.: A Lagrangian approach for integrated network design and routing in IP networks. In: Proceedings of International Network Optimization Conference (INOC 2003), Evry/Paris (2003) 107–113
[5] Bley, A., Koch, T.: Integer programming approaches to access and backbone IP-network planning. Technical Report ZR-02-41, Konrad-Zuse-Zentrum für Informationstechnik Berlin (2002)
[6] Farago, A., Szentesi, A., Szviatovski, B.: Allocation of administrative weights in PNNI. In: Proceedings of Networks 1998. (1998) 621–625
[7] Ben-Ameur, W., Gourdin, E., Liau, B., Michel, N.: Optimizing administrative weights for efficient single-path routing. In: Proceedings of Networks 2000. (2000)
[8] Prytz, M.: On Optimization in Design of Telecommunications Networks with Multicast and Unicast Traffic. PhD thesis, Royal Institute of Technology, Stockholm, Sweden (2002)
[9] Dinitz, Y., Garg, N., Goemans, M.: On the single source unsplittable flow problem. Combinatorica **19** (1999) 1–25
[10] Kolliopoulos, S., Stein, C.: Improved approximation algorithms for unsplittable flow problems. In: Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, Miami Beach FL. (1997) 426–435
[11] Skutella, M.: Approximating the single source unsplittable min-cost flow problem. Mathematical Programming **91** (2002)
[12] Garey, M., Johnson, D.: Computers and intractability: A Guide to the Theory of NP-Completeness. Freeman and Company, New York (1979)
[13] Fortz, B., Thorup, M.: Increasing internet capacity using local search. Computational Optimization and Applications **29** (2004) 13–48
[14] Lorenz, D., Orda, A., Raz, D., Shavitt, Y.: How good can IP routing be? Technical Report 2001-17, DIMACS, Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, Princeton University, AT&T Bell Laboratries and Bellcore (2001)

Inventory and Facility Location Models with Market Selection

Retsef Levi[1,⋆], Joseph Geunes[2,⋆⋆], H. Edwin Romeijn[3,⋆⋆⋆], and David B. Shmoys[4,†]

[1] School of ORIE, Cornell University, Ithaca, NY 14853
rl227@cornell.edu

[2] Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611-6595
geunes@ise.ufl.edu

[3] Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611-6595
romeijn@ise.ufl.edu

[4] School of ORIE and Department of Computer Science, Cornell University Ithaca, NY 14853
shmoys@cs.cornell.edu

Abstract. We consider important generalizations of a wide class of traditional deterministic inventory and facility location models that we call *inventory/facility location models with market selection*. Instead of the traditional setting, we are given a set of *markets*, each specified by a sequence of demands and associated with a revenue. Decisions are made in two stages. We first make a decision of what markets to select, where all other markets are rejected. Next we have to construct a minimum-cost production plan (facility layout) to satisfy all of the demands of all the selected markets. The goal is to minimize the overall lost revenues of rejected markets and the production (facility openings and connection) costs. We show how to leverage existing approximation results for the traditional models to corresponding results for the counterpart models with market selection. More specifically, any LP based α−approximation for the traditional model can be leveraged to a $\frac{1}{1-e^{-\frac{1}{\alpha}}}$−approximation algorithm for the counterpart model with market selection. Our techniques are also applicable to an important class of covering problems.

1 Introduction

Traditional deterministic inventory theory provides streamlined optimization models where the goal is to satisfy a sequence of specified demands over a given

⋆ Research supported partially by a grant from Motorola and NSF grants CCR-9912422& CCR-0430682.

⋆⋆ Research supported partially by NSF grants DMI-0322715 and DMI-0355533.

⋆⋆⋆ Research supported partially by NSF grant DMI-0355533.

† Research supported partially by NSF grants CCR-9912422& CCR-0430682.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 111–124, 2005.

planning horizon with minimum overall cost, optimally balancing different costs in the supply chain. Recent trends in supply chain research and practice have led to a broadened perspective that considers not only decisions on the supply side but on the demand side as well. More specifically, the demands to which the supply chain must respond and commit to are not completely exogenous parameters, but may be influenced by endogenous decisions such as pricing, promotions, and other strategic marketing-based factors. In particular, a supplier should strive to optimally fit the *shape* of the demand to the supply chain's production capabilities. A fundamental aspect of these issues is the choice of markets (or customers) to which the supplier commits to sell commodities. That is, given products and a set of potential markets, which markets should the supplier target for sales? Clearly, this decision is impacted by the potential profitability of each market. While marketing models can be used to predict the potential *revenue* available in a market, economies of scale in production make it difficult to isolate the *cost* of serving a market, in particular, because this cost depends on the collective set of markets being served. Aiming to address such *market selection* decisions in the planning phase, we propose the following integrated market selection and production planning problem.

We consider generalizations to a wide class of traditional deterministic inventory models that capture the above mentioned issues. In these more general models, the input consists of a set of *markets* (or customers), where each is specified by a sequence of demands over the given planning horizon and is associated with a potential revenue. The decisions are made in two stages. First we decide which markets we are going to *select* and which ones will be *rejected*. By selecting a market we commit to satisfy *all* of its demands over the horizon. Rejecting a market implies that we can ignore all of its demands, but we lose its potential revenue. Once a subset of markets is selected, we need to construct a minimum cost production plan to satisfy all of the demands of the selected markets on time. As in the traditional inventory models the production cost usually consists of a *fixed ordering cost* that is incurred in each period an order is placed, and of *holding cost* for carrying excess inventory from period to period. Our goal is to minimize the overall cost, namely the *lost revenue* incurred by market rejection and the *production cost* of satisfying the demands of the selected markets. We call this new class of extended models *inventory models with market selection*.

All of the inventory models we consider in this paper can be formulated as facility location type integer programs that admit strong LP relaxations. We show how to modify these LP's to capture the more general models with market selection. We then describe general LP-rounding and primal-dual frameworks that leverage any LP-based approximation algorithm for the traditional inventory models to a corresponding algorithm for its counterpart with market selection. This results in a stream of approximation algorithms for these more general market selection models that have constant worst-case performance guarantees. That is, for any instance of the specific market selection problem, the algorithm provides a solution with cost at most constant multiple of the optimal cost.

The Models. Following are the details of some of the models to which our LP-rounding and primal-dual frameworks are applicable. In the *joint replenishment problem with market selection*, we consider a set of m markets, denoted by $\mathcal{M}$. Each market $j \in \mathcal{M}$ is specified by a sequence of demands for N commodities over a finite planning horizon of T periods and is associated with a potential revenue, r_j. The demands required by market j for item i are denoted by $d^j_{i1}, \ldots, d^j_{iT}$ (for $i = 1, \ldots, N$). The production cost structure is identical to the traditional joint replenishment problem (JRP). Each order incurs a *fixed joint ordering cost*, K_0, regardless of the combination of items being ordered. In addition, for each item included in the order, we incur a *fixed item ordering cost*, K_i, regardless of the number of units being ordered. The fixed ordering cost is balanced by the *holding cost* for carrying excess inventory over periods. The holding cost structure follows [13], i.e., for each $i = 1, \ldots, N$, $t = 1, \ldots, T$ and $s \leq t$, we have a parameter h^i_{st} that denotes the per unit cost of providing item i in period t from period $s \leq t$. The parameters h_{ist} are assumed to be non-negative and non-increasing in s. Per unit ordering cost can be incorporated into the h parameters. This generalizes the traditional holding cost structure, where we have a parameter $h_{it} \geq 0$ for each i and t, which denotes the per unit cost to hold inventory from period t to $t + 1$.

The *single item lot-sizing problem with market selection* is a special case of the JRP problem above, where we have only one commodity, i.e., $N = 1$. As a result there is no point to consider separate joint and item ordering cost, and instead we have just time dependent fixed ordering cost K_s (for each $s = 1, \ldots, T$).

The *one-warehouse multi-retailer (OWMR) problem with market selection* is a generalization of the JRP problem, where we assume that each item corresponds to a different retailer, and the retailers are supplied by a central warehouse that can hold inventory. We follow the general holding cost structure described in [18, 19].

In the *assembly problem with market selection*, each market is again specified by a sequence of T demands for a single commodity (end-product). However, the end-product is assembled from N components (see [13, 14] for details).

We will also consider some of these models with the additional constraints that orders are placed in *batches*, each with a given capacity. For each additional batch ordered, we incur an additional fixed ordering cost. These constraints are usually called *soft capacity constraints*.

In addition, our techniques are applicable to several market selection variants of facility location problems. This includes the classical uncapacitated metric facility location problem and its variants with soft capacities, and with service installation costs (see [22] for details).

The goal in all of these models is to first select a subset of the markets, and then construct a minimum cost production plan (or facility layout) for all of the demands of the selected markets, such that the overall cost incurred by market rejection and the production (facility layout and connection) plan is minimized.

Related Literature. The traditional inventory models mentioned above, have attracted the attention of many researchers throughout the years. The single item

lot-sizing problem can be solved to optimality using dynamic programming [23], including the case with soft capacity constraints [17]. The JRP and the OWMR problems are known to be NP-hard [1]. The assembly problem with general holding cost structure as in [13, 14] is also known to be NP-hard, and it is a long standing open problem whether or not it is NP-hard in the case of traditional holding cost. In several recent papers [13, 14, 18, 19], Levi, Roundy and Shmoys have developed LP-based approximation algorithms for these inventory models. In particular, they have provided a primal-dual 2-approximation algorithm for the JRP and the assembly problem, and an LP-rounding 2.398-approximation algorithm for the OWMR problem.

The facility location problem is known to be NP-hard, and there is a huge body of literature on approximation algorithms for this problem. Mahdian, Ye and Zhang have the currently best known result, an LP-based 1.52-approximation algorithm [15] (we refer the reader to [20] for a survey on approximation results on this problem).

There are several other models, such as the *prize-collecting travelling salesman problem* [2, 6], *prize-collecting Steiner tree* [10, 9] and *multi-processor scheduling with rejection* [4], where classical combinatorial optimization problems were extended to consider *rejection/selection* decisions. However, in all of these models the rejection /selection decisions are made independently for each element (node or a job). Our market selection models are different in that the rejection/selection is made with respect to a collection of elements as a set. We note that in their full generality our techniques and results are applicable to the market selection variants of the prize-collecting problems mentioned above, as well as to market selection variants of additional covering problems.

In [8], Geunes, Romeijn and Taaffe have considered a single item, single location *order selection model* where each market is specified by a single demand in a single period; they have provided an optimization algorithm based on dynamic programming. In this paper, we consider multi-item, multi-stage and capacitated inventory models with more complex selection decisions that are made with respect to markets specified over the time horizon rather than in a single period.

Our approach, when applied to the metric facility location problem, generalizes the model of *facility location with penalties* discussed by Charikar, Khuller, Mount and Narasimhan in [7]. They have also considered a model where the rejection/selection decision is made per demand point independently rather than with respect to subsets of demands. They have provided a 3-approximation algorithm for this model, extending the novel primal-dual algorithm of Jain and Vazirani [11] for the facility location problem. For the more general model considered in this paper, we provide a significantly better approximation factor of 2.075 using LP-rounding. We also provide a 2.542-approximation algorithm for the market selection variant of the facility location problem with soft capacities. However, our LP-rounding framework is not applicable to some of the facility location models they have considered.

Our Techniques and Results. Our LP-rounding framework builds on known facility location type LP relaxations for the traditional models. We modify the LP's to capture the corresponding market selection problems by adding a variable for each market that indicates whether it is rejected or not. If we think about this variable as representing a special 'order' (or facility), then we either serve all of the demands of the corresponding market by this special 'order' and lose the revenue of the market, or we have to assign each one of its demands to a regular order (facility). We solve the *market selection LP* relaxation and consider its optimal solution. The rounding algorithm is conceptually very simple. For some threshold $\frac{1}{\beta}$ (for some $\beta \geq 1$), we reject all the markets that the LP fractional solution rejects by more than $\frac{1}{\beta}$. Clearly, this gives a bound on the rejection cost that our solution incurs. The main observation is that once we decide on the selected markets, then the market selection LP is reduced to the *traditional LP* for the inventory (facility location) model, for an instance defined by the selected markets aggregated. We now construct the production plan (facility openings) for these markets using existing LP-based α-approximation algorithm for the traditional inventory (facility location) model. It is clear that the cost incurred is at most α times the optimal value of the (reduced) traditional LP. Finally, observe that if a certain market is fractionally rejected to an amount less than $\frac{1}{\beta}$, then the fractional production plan satisfies at least $1-\frac{1}{\beta}$ of each of its demands. Therefore, we can scale the fractional solution that corresponds to the selected markets by no more than $1-\frac{1}{\beta}$ to get a feasible fractional solution to the reduced traditional LP. This bounds our production cost to be at most $\alpha/(1-\frac{1}{\beta})$ times the optimal fractional production cost of the market selection LP. Optimizing deterministically over β we can now leverage any existing LP-based α-approximation algorithm for the traditional inventory problem to an $(\alpha+1)$-approximation algorithm for the corresponding market selection counterpart. However, drawing $\frac{1}{\beta}$ at random from the appropriate distribution, as described by Goemans in [9], provides a randomized algorithm with improved expected guarantees. Using derandomization we can now leverage any α-approximation into a deterministic $\frac{1}{1-e^{-\frac{1}{\alpha}}}$ guarantee. As a by product we also show that if the traditional model admits a strong LP relaxation so does its market selection counterpart. We note that somewhat similar ideas were used independently in [21] in the context of 2-stage stochastic models.

In addition, we show how to extend the primal-dual framework described in [13, 14] to provide combinatorial approximation algorithms for the market selection variants of the single item lot-sizing problem, the JRP problem and the assembly problem. We provide worst-case guarantees of 2, 3 and 3, respectively.

The rest of the paper is organized as follows. In Section 2, we start with the illustrative simple example of the single item lot-sizing problem. Then in Section 3, we describe the general LP-rounding framework and the approximation results it implies. Finally, in Section 4, we briefly discuss how to extend the primal-dual framework of [13, 14].

2 The Single Item Lot-Sizing Problem - An Illustrative Example

In this section, we will use the illustrative simple example of the single item lot-sizing problem with market selection to demonstrate the underlying techniques of our LP-rounding framework.

Next we describe a facility location type LP-relaxation of this selection problem that is based on the LP relaxation of its traditional counterpart. We then show how to round the optimal solution of this LP into a feasible solution to the market selection variant with cost at most 1.582 times the optimal cost.

The LP. The LP is as follows:

$$\text{minimize} \quad \sum_{j=1}^{m} z_j r_j \; + \; \sum_{s=1}^{T} y_s K_s \; + \sum_{j=1}^{m}\sum_{t=1}^{T}\sum_{s=1}^{t} x_{st}^j H_{st}^j \tag{P}$$

$$\text{subject to} \quad \sum_{s=1}^{t} x_{st}^j \; + \; z_j = 1, \qquad j = 1,\ldots,m,\ t = 1,\ldots,T, \tag{1}$$

$$x_{st}^j \le y_s, \qquad j = 1,\ldots,m,\ t = 1,\ldots,T, \quad s = 1,\ldots,t, \tag{2}$$

$$x_{st}^j,\ y_s \ge 0, \qquad j = 1,\ldots,m,\ s = 1,\ldots,T, \quad t = s,\ldots,T. \tag{3}$$

The *selection variable* z_j is equal to 1 if we decided to reject market j and 0 otherwise. The *assignment variable* x_{st}^j indicates whether the demand of market j in period t, d_t^j, was provided from period s. Correspondingly, we let H_{st}^j be the overall cost of providing the demand d_t^j from period s, i.e., $H_{st}^j = h_{st} d_t^j$. The *order variable* y_s indicates whether an order was placed in period s. The *assignment constraint* (1) ensures that each demand in period t of a selected market is satisfied from some time period $s \le t$. Note that if we have selected market j and set $z_j = 0$, then for each period t, constraint (1) implies that $\sum_{s \le t} x_{st}^j = 1$. Hence, all of the demands of market j must be satisfied on time. Conversely, if $z_j = 1$, then we do not have to satisfy any of the demands of market j, and in turn we incur a cost of r_j. The *order constraint* (2) ensures that no demand can be provided from period s without placing an order in period s. It is straightforward to verify that the induced integer programm provides a correct formulation for the single item lot-sizing problem with market selection, and therefore, the optimal solution of the LP-relaxation provides a lower bound on the cost of any feasible solution to the problem.

Observe that once we decide on the selected markets (i.e., assign zero/one value to each variable z_j), then the above LP is reduced to an LP of a traditional single item lot-sizing problem defined by the selected markets. The demand in period t is equal to the aggregated demands d_t^j over all markets $j \in \mathcal{M}$ that were selected (i.e., the demand in period t is equal to $\sum_{j \in \mathcal{M}} d_t^j$). Moreover, it is well known that this LP always provides an integer solution i.e., it has an

integrality property (for different proofs see [12, 3, 5, 13]). However, we note that the market selection LP, (P) above, does not have an integrality property (i.e., for some instances the optimal solution is fractional).

A Rounding Algorithm. Let $(\hat{z}, \hat{x}, \hat{y})$ be the optimal solution of (P) above. We next describe an extremely simple procedure to round this optimal fractional solution into a feasible solution for the single item lot-sizing problem with market selection, denoted by $(\bar{z}, \bar{x}, \bar{y})$, and then show that its cost is always at most twice the optimal cost.

Let $\mathcal{Z}_{\frac{1}{2}} := \{j \in \mathcal{M} : \hat{z}_j \geq \frac{1}{2}\}$, be the set of all markets that are rejected by 'amount' of at least $\frac{1}{2}$ in the fractional optimal solution. Let $\mathcal{Z}_S := \mathcal{M} \setminus \mathcal{Z}_{\frac{1}{2}}$. We now reject all the markets $j \in \mathcal{Z}_{\frac{1}{2}}$ and select all other markets, i.e., we set $\bar{z}_j = 1$ if and only if $\hat{z}_j \geq \frac{1}{2}$. Next consider the single item lot-sizing problem defined by the selected markets, i.e., by the markets $j \in \mathcal{Z}_S$, where again the demand in period t is equal to $\sum_{j \in \mathcal{Z}_S} d_t^j$. This problem can be efficiently solved to optimality using dynamic programming (see [23] for details). We take this solution as our production plan. This concludes the description of the algorithm.

Analysis. We start with a lemma that bounds the rejection cost incurred in $(\bar{z}, \bar{x}, \bar{y})$. The proof is rather straightforward and follows from the construction of the algorithm.

Lemma 1. *The rejection cost of the solution $(\bar{z}, \bar{x}, \bar{y})$ is at most twice the rejection cost of the optimal fractional solution $(\hat{z}, \hat{x}, \hat{y})$, i.e., $\sum_{j=1}^{m} \bar{z}_j \leq 2 \sum_{j=1}^{m} \hat{z}_j$.*

To complete the analysis, it is enough to show that the production cost of the solution $(\bar{z}, \bar{x}, \bar{y})$ is at most twice the production cost of the solution $(\hat{z}, \hat{x}, \hat{y})$.

Lemma 2. *The production cost of the solution $(\bar{z}, \bar{x}, \bar{y})$ is at most twice the production cost of $(\hat{z}, \hat{x}, \hat{y})$: $\sum_{s=1}^{T} \bar{y}_s K_s + \sum_{j=1}^{m} \sum_{t=1}^{T} \sum_{s=1}^{t} \bar{x}_{st}^j H_{st} \leq 2(\sum_{s=1}^{T} \hat{y}_s K_s + \sum_{j=1}^{m} \sum_{t=1}^{T} \sum_{s=1}^{t} \hat{x}_{st}^j H_{st})$.*

Proof. Recall that once we decide on the selected markets, then the problem and the LP are reduced to a traditional lot-sizing problem defined by the markets in $\mathcal{Z}_S$. Moreover, since the facility location LP of the lot-sizing problem has an integrality property, we know that the optimal value of that LP is equal to the value of the optimal solution achieved by applying dynamic programming to this problem. It is then enough to observe a feasible solution to this reduced LP that has cost at most twice the production cost in $(\hat{z}, \hat{x}, \hat{y})$. We next describe how to construct such a solution, denoted by (x, y).

For each $j \in \mathcal{Z}_{\frac{1}{2}}$ we set $x_{st}^j = 0$ for $s \leq t$. Now for each $j \in \mathcal{Z}_S$ we set $x_{st}^j := \frac{\hat{x}_{st}^j}{1 - \hat{z}_j}$. Finally, for each $s = 1, \ldots, T$ we set $y_s := \max\{\min_{j \in \mathcal{Z}_S}(\frac{\hat{y}_s}{1 - \hat{z}_j}), 1\}$. It is readily verified that (x, y) constructed above is a feasible fractional solution for the LP of the lot-sizing problem defined by the markets in $\mathcal{Z}_S$.

Observe that for every $j \in \mathcal{Z}_S$ we have $1 - \hat{z}_j > \frac{1}{2}$. Hence, for each $j \in \mathcal{Z}_S$ and $s \leq t$ we have $x_{st}^j \leq 2\hat{x}_{st}^j$ and for each $s = 1, \ldots, T$ we have $y_s \leq 2\hat{y}_s$. Now

by construction of the algorithm $(\bar{x}, \bar{y})$ provides an optimal solution to the LP of the lot-sizing problem defined by the markets in $\mathcal{Z}_S$. We get that,

$$\sum_{s=1}^{T} \bar{y}_s K_s + \sum_{j=1}^{m} \sum_{t=1}^{T} \sum_{s=1}^{t} \bar{x}_{st}^j H_{st} \leq \sum_{s=1}^{T} y_s K_s + \sum_{j=1}^{m} \sum_{t=1}^{T} \sum_{s=1}^{t} x_{st}^j H_{st} \leq 2\left(\sum_{s=1}^{T} \hat{y}_s K_s + \sum_{j=1}^{m} \sum_{t=1}^{T} \sum_{s=1}^{t} \hat{x}_{st}^j H_{st}\right).$$

This concludes the proof of the lemma.

As a corollary of Lemmas 1 and 2 we get the following theorem.

Theorem 1. *The algorithm provides a 2-approximation algorithm for the lot-sizing problem with market selection, i.e., the cost* $(\bar{z}, \bar{x}, \bar{y})$ *is at most twice the optimal cost.*

2.1 Randomized Algorithm

Next we briefly discuss how randomized rounding yields an improved approximation algorithm. Instead of using the threshold $\frac{1}{2}$, we follow the ideas in [9], and choose a threshold $\frac{1}{\beta}$ uniformly at random from $(0, \delta]$, where $0 < \delta \leq 1$ (the value of δ will be specified later). Once $\frac{1}{\beta}$ is chosen we proceed in the same way described above, i.e., we reject all markets $j \in \mathcal{M}$ with $\hat{z}_j \geq \frac{1}{\beta}$, and construct a minimum-cost production plan for the selected markets (using dynamic programming).

Following are several lemmas that establish the expected performance guarantee of the randomized algorithm described above.

Lemma 3. *For each market* $j \in \mathcal{M}$*, the probability that market* j *is rejected is at most* $\frac{\hat{z}_j}{\delta}$*, i.e., the expected rejection cost of the solution is at most* $\frac{\sum_j \hat{z}_j r_j}{\delta}$.

Proof. For each market j with $\hat{z}_j \leq \delta$ the claim follows trivially. For each market j with $\hat{z}_j > \delta$, we reject the market with probability 1, however, $\frac{\hat{z}_j}{\delta} > 1$. This concludes the proof.

Lemma 4. *The expected production cost of the solution is at most* $\frac{\ln(\frac{1}{1-\delta})}{\delta}\left(\sum_{s=1}^{T} \hat{y}_s K_s + \sum_{j=1}^{m} \sum_{t=1}^{T} \sum_{s=1}^{t} \hat{x}_{st}^j H_{st}\right)$.

Proof. By similar arguments to Lemma 2 above, it is readily verified that, for each value of $\frac{1}{\beta}$, the production cost is at most $\frac{1}{1-\frac{1}{\beta}}$ times the production cost of the optimal LP solution, $(\hat{z}, \hat{z}, \hat{y})$. Taking the integral $\frac{1}{\delta} \int_0^{\delta} \frac{1}{1-\frac{1}{\beta}} d(\frac{1}{\beta})$ gives the required result.

Setting δ equal to $1 - e^{-1}$ provides and a solution with expected cost at most 1.582 times the optimal cost. Observe that the solution constructed by the

algorithm is uniquely determined by the set of rejected markets. Hence, there are at most m relevant values of $\frac{1}{\beta}$ that will yield distinct solutions. We can then derandomize the algorithm by enumerating only the m relevant values of $\frac{1}{\beta}$ and get the following theorem.

Theorem 2. *There exists a 1.582-approximation algorithm for the market selection variant of the lot-sizing problem.*

3 General LP-Rounding Framework

In this section, we will generalize the example discussed in Section 2 above, and propose a general LP-rounding framework that can be applied to market selection variants of a wide class of classical deterministic inventory and facility location models. The ingredients for our LP-rounding framework are a facility location type LP relaxation and an LP-based α-approximation algorithm for the traditional model without market selection. We show how, given these ingredients, one can leverage the α-approximation algorithm into an $\frac{1}{1-e^{-\frac{1}{\alpha}}}$-approximation algorithm for the corresponding counterpart model with market selection.

Our algorithms can use any facility location type LP with *assignment variables*, x, and *order variables*, y, and with *assignment constraints* (e.g., constraint (1) above), *ordering constraints* (e.g., constraint (2) above) and possibly *soft capacity constraints*. Here the order variables y indicate how many batches were ordered in each order, where each batch incurs an additional fixed ordering cost. The number of batches installed dictates the capacity of the order, or in other words, how many units of demand can be satisfied by this order. For the lot-sizing example discussed in Section 2, but with soft capacity constraints, we would have the constraint, $\sum_{t \geq s} \sum_{j=1}^{m} x_{st}^j d_t^j \leq y_s U_s$, for each period $s = 1, \ldots, T$, where U_s is the batch capacity associated with the order in period s. By an LP-based α-approximation algorithm we mean an algorithm that constructs a solution with cost guaranteed to be at most α times the optimal LP value.

We now state and prove a general theorem that relates the approximation results for traditional deterministic inventory (and facility location models) to approximation results for their corresponding counterparts with market selection.

Theorem 3. *Consider a deterministic inventory or a facility location model that can be formulated as a facility location type integer program with assignment and order variables, and with assignment constraints, order constraints and possibly soft capacity constraints. Also assume that the model has an LP-based α-approximation. Then there exists an $(\alpha + 1)$-approximation algorithm for the counterpart of this model with market selection.*

Proof. We first adjust the LP relaxation for the traditional model to capture the counterpart model with market selection. Let $\mathcal{M}$ be again the set of m markets.

As before we let the selection variable z_j be equal 1 if we reject market j and 0 otherwise. For each demand of market j, we have a separate set of assignment variables (for each facilty/order that can serve it). Similarly there is an assignment constraint for each of its demands that includes the z_j variable. The order constraints and possibly the soft capacity constraints are adapted accordingly. Finally, we modify the objective function and add the market rejection part $\sum_{j=1}^{m} z_j r_j$. We now solve this LP and let $(\hat{z}, \hat{x}, \hat{y})$ be the respective optimal solution.

Now for some parameter $\beta > 1$ that will be determined later, let $\mathcal{Z}_{\frac{1}{\beta}} := \{j : \hat{z}_j \geq \frac{1}{\beta}\}$, i.e., the set of markets that are rejected in the market selection LP optimal solution by 'amount' at least $\frac{1}{\beta}$. Let $\mathcal{Z}_S$ be the set of all other markets (i.e., $\mathcal{Z}_S = \mathcal{M} \setminus \mathcal{Z}_{\frac{1}{\beta}}$). We now reject all the markets in $\mathcal{Z}_{\frac{1}{\beta}}$ and select all markets $j \in \mathcal{Z}_S$. Next consider the instance for the traditional variant of the model defined by the markets in $\mathcal{Z}_S$, where again, for each period (location) and each commodity, we aggregate the corresponding demands of all the selected markets. We then apply the α-approximation algorithm to the instance defined by $\mathcal{Z}_S$, and get a production plan (facility location solution).

It is straightforward to see that the rejection cost incurred by the solution is at most $\beta \sum_{j=1}^{m} \hat{z}_j r_j$. Also observe that by a similar scaling to the one described in Lemma 2, we can observe a feasible solution to the traditional LP defined by the markets in $\mathcal{Z}_S$ of cost at most $\frac{\beta}{\beta-1}$ times the cost of the (fractional) production plan in $(\hat{z}, \hat{x}, \hat{y})$. However, by our assumption, the cost of the production plan constructed by the algorithm, is at most α times the cost of the optimal solution to the above traditional LP defined by the markets in $\mathcal{Z}_S$. Therefore, it is at most $\alpha(\frac{\beta}{\beta-1})$ times the cost of the production plan in $(\hat{z}, \hat{x}, \hat{y})$. Setting $\beta = \alpha + 1$, we get that the cost of the solution is at most $\alpha + 1$ times the cost of $(\hat{z}, \hat{x}, \hat{y})$. This concludes the proof.

Similar to the lot-sizing case, if we choose $\frac{1}{\beta}$ at random uniformly from $(0, \delta]$ (where again $0 < \delta \leq 1$), then the expected rejection cost is at most $\frac{1}{\delta}$ the rejection cost in the optimal LP solution, $(\hat{z}, \hat{x}, \hat{y})$ (i.e., at most $\frac{\sum_j \hat{z}_j r_j}{\delta}$), and the expected production cost is at most $\frac{\alpha \ln(\frac{1}{1-\delta})}{\delta}$ times the production cost in $(\hat{z}, \hat{x}, \hat{y})$ (i.e., at most $\frac{\alpha \ln(\frac{1}{1-\delta})}{\delta}(\sum_{s=1}^{T} \hat{y}_s K_s + \sum_{j=1}^{m} \sum_{t=1}^{T} \sum_{s=1}^{t} \hat{x}_{st}^{j} H_{st})$). Setting δ equal to $1 - e^{-\frac{1}{\alpha}}$ guarantees that expected cost of the algorithm is at most $\frac{1}{1-e^{-\frac{1}{\alpha}}}$ times the optimal cost. By the same derandomization techniques, we achieve the same guarantee deterministically.

Approximation Results. The *single item lot-sizing problem with soft capacity constraints* can be solved optimally [17]. In [18], Levi, Roundy and Shmoys have showed that the integrality gap of the facility location type LP of this model is at most 2. We get the following theorem.

Theorem 4. *There exists a 2.542-approximation algorithm for the market selection single item lot-sizing problem with soft capacity constraints.*

We consider a sequence of LP-based approximation algorithms described in [13, 14, 18, 19], namely, a 2-approximation algorithms for the JRP and assembly problems, a 4-approximation algorithm for the JRP with soft capacities, a 2.398-approximation algorithm for the OWMR and a 4.769-approximation algorithm for the OWMR with soft capacities. We then conclude the following theorems.

Theorem 5. *There exist a 2.54-approximation algorithms for the market selection variants of the joint replenishment and the assembly problems, and a 4.521-approximation algorithm for market selection variant of the JRP with soft capacities.*

Theorem 6. *There exist a 2.993-approximation algorithm and a 5.287-approximation algorithm for the market selection variants of the OWMR and the OWMR with soft capacities, respectively.*

Finally, in [15, 16], Mahdian, Ye and Zhang have provided a 1.52-approximation algorithm and 2-approximation algorithm for the facility location problem and its variant with soft capacities, respectively. Both algorithms are LP-based. In [22], Shmoys, Swamy and Levi have provided a 6-approximation algorithm for the facility location problem with service installation costs. We again conclude a corresponding theorem.

Theorem 7. *There exist a 2.075, 2.542 and 6.514 approximation algorithms for the market selection variants of the uncapacitated facility location problem, the facility location problem with soft capacities and facility location with service installation costs.*

4 Extended Primal-Dual Framework

In [13, 14], Levi, Roundy and Shmoys have described a general primal-dual framework that solves the lot-sizing problem to optimality, and provides a 2-approximation for the JRP and assembly problem. In this section, we will briefly discuss how to modify their algorithms to provide combinatorial approximation algorithms (i.e., algorithms that do not require solving the LP) for the counterpart models with market selection. Using these modified algorithms, we provide performance guarantees of 2, 3 and 3 for these models, respectively. For simplicity and due lack of space, we next focus on the single item lot-sizing problem. Similar extensions exist for the JRP and assembly problem.

The following is the dual of (P) from Section 2.

$$\text{maximize} \quad \sum_{j=1}^{m}\sum_{t=1}^{T} b_t^j \tag{D}$$

$$\text{subject to} \quad b_t^j \le H_{st}^j + l_{st}^j, \qquad \begin{matrix} j = 1, \ldots, m, \ t = 1, \ldots, T. \\ s = 1, \ldots, t. \end{matrix} \tag{4}$$

$$\sum_{j=1}^{m}\sum_{t=s}^{T} l_{st}^{j} \le K_s, s = 1,\ldots,T, \tag{5}$$

$$\sum_{t=1}^{T} b_t^j \le r_j,\ j = 1,\ldots,m, \tag{6}$$

$$l_{st} \ge 0,\quad t = 1,\ldots,T,\ s = 1,\ldots,t. \tag{7}$$

The above dual program is very similar to the dual used in [13, 14], and the main difference is the additional constraint (6). As in [13, 14], the primal-dual algorithm works in two stages. We first construct a feasible dual solution, and then use this solution to construct a cheap feasible integer solution for the primal LP.

Constructing a Dual Solution. We associate a budget b_t^j with each demand point (j,t) (i.e., the required demand of market j in period t, d_t^j). We now start to increase the budget using a mechanism that is called *wave form*. Think of a wave the moves backward in time, and let τ be the indicator of the *wavefront* location. We initialize the wavefront variable τ to T. The algorithm consists of a series of iterations as the value of τ is (continuously) decreased through the interval $[T, 1]$. This parameter controls the values of the budgets b_t^j of each unfrozen demand point (j,t). The budget b_t^j is kept equal zero as long as $\tau > t$. Once $\tau \le t$ and until it is frozen, b_{it}^j is always equal $H_{\tau t}^j$. As the wave moves backward in time, we will *temporarily open orders* and *freeze* budgets (i.e., stop increasing then) of demand points; as the budgets are increased we identify the following events:

Event 1. When $\tau = s$ (for $s = T, T-1,\ldots,1$), we consider all unfrozen demand points (j,t) with $t \ge s$, and start increasing the variable l_{st}^j at the same rate as b_t^j, i.e., we keep $b_t^j = H_{\tau t}^j = H_{st}^j + l_{st}^j$; (Note that as the wavefront reaches s and the budget b_t^j increases to H_{st}^j the constraint (4) becomes tight).
Event 2. Suppose that for some s, we have that $\sum_{j=1}^{m}\sum_{t\ge s} l_{st}^j = K_s$. (Note that this means that we can no longer increase any variables l_{st}^j without violating the constraint (5)). We then temporarily open an order in period s, and freeze all unfrozen budgets of demand points (j,t) with $t \ge s$.
Event 3. Suppose that for some market j we have $\sum_{t=1}^{T} b_t^j = r_j$ (i.e., constraint (6) becomes tight). We then *mark* market j and freeze *all* the budgets of its demand points. This includes also budgets of demand points with $t < \tau$ that will be kept at zero even after τ will advance and cross time t.

We continue this procedure until all the budgets are frozen. Let $(\bar{b}, \bar{l})$ be the dual solution at the end of the first phase of the algorithm. It is straightforward to verify that this solution is feasible with respect to (D) above. Next we show how to use this solution to construct a feasible solution for the market selection problem.

Constructing an Integer Primal Solution. We reject all markets that were marked by event 3 above. We then construct a production plan for the rest of the markets

following a similar procedure to the one described in [13, 14]. Let $R = \{s_1 = 1 < s_2 < \cdots < s_p\}$ be the set of the time periods of all temporarily opened orders. For each $s \in R$, let $open(s)$ be the location of the wavefront when the order at s was temporarily opened. We say that the interval $[open(s), s]$ is the *shadow interval* of s. Furthermore, r and s in R are said to be *dependent* if and only if their shadow intervals intersect. We consider the periods s_i, $i = 1, \ldots, p$, in increasing order of s_i, and permanently open an order s_q whenever its associated shadow interval does not intersect the shadow interval of any earlier s_i, $i = 1, \ldots, q-1$, that has already been permanently opened. The set of permanently opened orders provides a feasible production plan, where each demand point of a selected market is satisfied from the latest possible opened order.

Analysis. The analysis follows along the lines of [13, 14]. By similar arguments to the ones used there, we can show that it is possible to 'pay' for the cost of the production plan of the solution, using every budget $\bar{b}_t^j$ at most once. In addition, it is clear that the rejection cost can be 'paid' with the budgets associated with the demands of the rejected markets. Therefore, the total cost of the solution can be 'paid' using each budget $\bar{b}_t^j$ at most twice. For the JRP and assembly problem the modification in the algorithm is similar. Here it can be shown that the production cost can be 'paid' using each budget at most twice, and the rejection cost is again 'paid' by the budgets of the rejected markets. We then conclude the following theorem.

Theorem 8. *The modified primal-dual framework provides approximation algorithms with worst-case performance guarantees of 2, 3 and 3, respectively, for the market selection variants of the lot-sizing, the JRP and the assembly problem.*

References

1. E. Arkin, D. Joneja, and R. Roundy. Computational complexity of uncapacitated multi-echelon production planning problems. *Operations Research Letters*, 8:61–66, 1989.
2. E. Balas. The prize collection travelling salesman problem. *Networks*, 19:621–636, 1989.
3. I. Bárány, T. J. Van Roy, and L. A. Wolsey. Uncapacitated lot-sizing: the convex hull of solutions. *Mathematical Programming Study*, 22:32–43, 1984.
4. Y. Bartal, S. Leonardi, A. M. Spaccamela, J. Sgall, and L. Stougie. Multiprocessor scheduling with rejection. *SIAM Journal on Discrete Mathematics*, 13:64–78, 2000.
5. D. Bertsimas, C. Teo, and R. Vohra. On dependent randomized rounding algorithms. *Operations Research Letters*, 25:105–114, 1999.
6. D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Mathematical Programming*, 59:413–420, 1993.
7. M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 642–651, 2001.

8. J. Geunes, H. E., Romeijn, and K. Taaffe. Requirements planning with pricing and order selection flexibility. To appear in Operations Research, 2004.
9. M. X. Goemans. Approximate solutions of hard combinatorial problems, 1996. Lecture notes in the schoolof ASHCOMP.
10. M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296–317, 1995.
11. K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM 48*, pages 274–296, 2001.
12. J. Krarup and O. Bilde. Plant location, set covering and economic lot sizing: an O(mn) algorithm for structural problems. In *Numerische Methoden Bei Optimierungsaufgaben*, volume 3, pages 155–180, 1977.
13. R. Levi, R. O. Roundy, and D. B. Shmoys. Primal-dual algorithms for deterministic inventory problems. Technical Report TR1042, ORIE Department, Cornell University, 2004. Submitted.
14. R. Levi, R. O. Roundy, and D. B. Shmoys. Primal-dual algorithms for deterministic inventory problems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 353–362, 2004.
15. M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location. In *Proceedings of 6th APPROX*, pages 229–242, 2002.
16. M. Mahdian, Y. Ye, and J. Zhang. A 2-approximation algorithm for the soft-capacitated facility location problem. In *Proceedings of 7th APPROX*, pages 129–140, 2003.
17. Y. Pochet and L. A. Wolsey. Lot-sizing with constant batches: *Formulation* and valid inequalities. *Mathematics of Operations Research*, 18:767–785, 1993.
18. R.Levi, R. Roundy, and D. Shmoys. A constant approximation algorithm for the one-warehouse multi-retailer problem. Technical Report TR1408, ORIE Department, Cornell University, 2004. Submitted, extended abstract will appear in SODA 2005.
19. R.Levi, R. Roundy, and D. Shmoys. A constant approximation algorithm for the one-warehouse multi-retailer problem (Extended Abstract). In *Proceedings of the 16th Annual SIAM-ACM Symposium on Discrete Algorithms*, 2005.
20. D. B. Shmoys. The design and analysis of approximation algorithms: facility location as a case study. In S. Hosten, J. Lee, and R. Thomas, editors, *Trends in Optimization. AMS Proceedings of Symposia in Applied Mathematics*, 2004. To appear.
21. D. B. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *Proceedings of the 45th Annual IEEE Symposium on the Foundations of Computer Science, 2004.*, pages 228–237, 2004.
22. D. B. Shmoys, C. Swamy, and R. Levi. Facility location with service installation costs. In *Proceedings of the 15th Annual SIAM-ACM Symposium on Discrete Algorithms*, pages 1081–1090, 2004.
23. H. M. Wagner and T. M. Whitin. Dynamic version of the economic lot sizing model. *Management Science*, 5:89–96, 1958.

On Approximating Complex Quadratic Optimization Problems via Semidefinite Programming Relaxations*

Anthony Man–Cho So[1], Jiawei Zhang[2], and Yinyu Ye[3]

[1] Department of Computer Science, Stanford University, Stanford, CA 94305, USA
manchoso@cs.stanford.edu
[2] Department of Information, Operations, and Management Sciences,
Stern School of Business, New York University,
New York, NY 10012, USA
jzhang@stern.nyu.edu
[3] Department of Management Science and Engineering and, by courtesy,
Electrical Engineering, Stanford University,
Stanford, CA 94305, USA
yinyu-ye@stanford.edu

Abstract. In this paper we study semidefinite programming (SDP) models for a class of discrete and continuous quadratic optimization problems in the complex Hermitian form. These problems capture a class of well–known combinatorial optimization problems, as well as problems in control theory. For instance, they include MAX–3–CUT where the Laplacian matrix is positive semidefinite (in particular, some of the edge weights can be negative). We present a generic algorithm and a unified analysis of the SDP relaxations which allow us to obtain good approximation guarantees for our models. Specifically, we give an $(k\sin(\frac{\pi}{k}))^2/(4\ )$–approximation algorithm for the discrete problem where the decision variables are k–ary and the objective matrix is positive semidefinite. To the best of our knowledge, this is the first known approximation result for this family of problems. For the continuous problem where the objective matrix is positive semidefinite, we obtain the well–known $\ /4$ result due to [2], and independently, [12]. However, our techniques simplify their analyses and provide a unified framework for treating these problems. In addition, we show for the first time that the integrality gap of the SDP relaxation is precisely $\ /4$. We also show that the unified analysis can be used to obtain an $O(1/\log\ )$–approximation algorithm for the continuous problem in which the objective matrix is not positive semidefinite.

1 Introduction

Following the seminal work of Goemans and Williamson [6], there has been an outgrowth in the use of semidefinite programming (SDP) for designing approxi-

* This research is supported in part by NSF grant DMS–0306611.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 125–135, 2005.

mation algorithms. Recall that an α–approximation algorithm for a problem $\mathscr{P}$ is a polynomial–time algorithm such that for every instance I of $\mathscr{P}$, it delivers a solution that is within a factor of α of the optimum value [5]. It is well–known that SDPs can be solved in polynomial time (up to any prescribed accuracy) via interior–point algorithms (see, e.g., [10]), and they have been used very successively in the design of approximation algorithms for, e.g., graph partitioning, graph coloring, and quadratic optimization problems [4, 11].

In this paper, we consider a class of discrete and continuous quadratic optimization problems in the complex Hermitian form. Specifically, we consider the following problems:

$$\begin{array}{ll} \text{maximize} & z^H Q z \\ \text{subject to} & z_j \in \{1, \omega, \ldots, \omega^{k-1}\} \qquad j = 1, 2, \cdots, n \end{array} \tag{1}$$

and

$$\begin{array}{ll} \text{maximize} & z^H Q z \\ \text{subject to} & |z_j| = 1 \qquad j = 1, 2, \cdots, n \\ & z \in C^n \end{array} \tag{2}$$

where $Q \in C^{n \times n}$ is a Hermitian matrix, ω is the principal k–th root of unity, and z^H denotes the conjugate transpose of the complex vector $z \in C^n$. The difference between (1) and (2) lies in the values that the decision variables are allowed to take. In problem (1), we have discrete decision variables, and such variables can be conveniently modelled as roots of unity. On the other hand, in problem (2), the decision variables are constrained to lie on the unit circle, which is a continuous domain. Such problems arise from many applications. For instance, the MAX–3–CUT problem where the Laplacian matrix is positive semidefinite can be formulated as an instance of (1). On the other hand, (2) arises from the study of robust optimization as well as control theory [9, 2].

It is known that both of these problems are NP–hard, and thus we will settle for approximation algorithms. Previously, various researchers have considered SDP relaxations for (1) and (2). However, approximation guarantee is known only for the continuous problem [2, 12], and to the best of our knowledge, no such guarantees are known for the discrete problem.

Our main contribution is to present a generic algorithm and a unified treatment of the two seemingly very different problems (1) and (2) using their natural SDP relaxations, and to give the first known approximation result for the discrete problem. Specifically, we are able to achieve an $(k \sin(\frac{\pi}{k}))^2/(4\pi)$–approximation ratio for the discrete problem[1]. As a corollary, we obtain an 0.537–approximation algorithm for the MAX–3–CUT problem where the Laplacian matrix is positive semidefinite. This should be contrasted with the 0.836–approximation algorithm of Goemans and Williamson [7] for MAX–3–CUT with *non–negative* edge weights. For this particular case, our result might be seen

[1] Recently, Zhang and Huang [13] have informed us that by extending their analysis in [12], they are able to obtain the same approximation ratio for the discrete problem.

as a generalization of Nesterov's result [8] that gives an $2/\pi$–approximation for the MAX–CUT problem where the Laplacian matrix is positive semidefinite.

For the continuous problem, our analysis also achieves the $\pi/4$ guarantee of [2, 12]. However, our analysis is simpler than that of [2, 12], and it follows the same framework as that of the discrete problem. Moreover, we give a tight example showing that the integrality gap of the SDP relaxation is precisely $\pi/4$. In addition, we show that the unified analysis can be used to obtain an $O(1/\log n)$–approximation algorithm for the continuous problem in the case where the objective matrix is not positive semidefinite. This result also provides an alternative analysis of the algorithm by Charikar and Wirth [3] for the (real) quadratic optimization problem.

One apparent difficulty in analyzing SDP relaxation–based algorithms for problems (1) and (2) is that the usual Goemans–Williamson analysis [6, 7] (and its variants thereof) only provides a term–by–term estimate of the objective function and does not provide a global estimate. Although global techniques for analyzing (real) SDP relaxations exist [8], it is not clear how they can be applied to our problems. Our analysis is mainly inspired by a recent result of Alon and Naor [1], who proposed three different methods for analyzing (real) SDP relaxations in a global manner using results from functional analysis. One of these methods uses averaging with Gaussian measure and the simple fact that $\sum_{i,j} q_{ij}(v_i \cdot v_j) \geq 0$ if the matrix $Q = (q_{ij})$ is positive semidefinite and $v_i \cdot v_j$ is the inner product of two vectors v_i and v_j in some Hilbert space. Our results for (1) and (2) in the case where Q is positive semidefinite are motivated by this method. Although the assumption that Q is positive semidefinite is essential to make the analyses go through, we manage to analyze our algorithm in a unified way for the case where Q is not positive semidefinite as well.

2 Complex Quadratic Optimization

Let $Q \in C^{n \times n}$ be a Hermitian matrix. Given an integer $n \geq 1$, consider the following discrete quadratic optimization problem:

$$\begin{array}{ll} \text{maximize} & z^H Q z \\ \text{subject to} & z_j \in \{1, \omega, \ldots, \omega^{k-1}\} \qquad j = 1, 2, \cdots, n \end{array} \tag{3}$$

where ω is the principal k–th root of unity. We note that as k goes to infinity, the discrete problem (3) becomes a continuous optimization problem:

$$\begin{array}{ll} \text{maximize} & z^H Q z \\ \text{subject to} & |z_j| = 1 \qquad j = 1, 2, \cdots, n \\ & z \in C^n \end{array} \tag{4}$$

Although problems (3) and (4) are quite different in nature, the following *complex* semidefinite program provides a relaxation for both of them:

$$\begin{array}{lll} \text{maximize} & Q \bullet Z & \\ \text{subject to} & Z_{jj} = 1 & j = 1, 2, \cdots, n \\ & Z \succeq 0 & \end{array} \tag{5}$$

We use w_{SDP} to denote the optimal value of the SDP relaxation (5).

Our goal is to get a near optimal solution for problem (3) and (4). Below we present a generic algorithm that can be used to solve both (3) and (4). Our algorithm is quite simple, and it is similar in spirit to the algorithm of Goemans and Williamson [6, 7].

Algorithm

STEP 1. Solve the SDP relaxation (5) and obtain an optimal solution Z^*. Since Z^* is positive semidefinite, we can obtain a Cholesky decomposition $Z^* = VV^H$, where $V = (v_1, v_2, \cdots, v_n)$.

STEP 2. Generate two independent normally distributed random vector $x \in R^n$ and $y \in R^n$ with mean 0 and covariance matrix $\frac{1}{2}I$. Let $r = x + yi$.

STEP 3. For $j = 1, 2, \cdots, n$, let $\hat{z}_j = f(v_j \cdot r)$, where the function $f(\cdot)$ depends on the structure of the problem and will be fixed later. Let $\hat{z} = (\hat{z}_1, \hat{z}_2, \cdots, \hat{z}_n)$ be the resulting solution.

In order to prove the performance guarantee of our algorithm, we are interested in analyzing the quantity:

$$\hat{z}^H Q \hat{z} = Q \bullet \hat{z}\hat{z}^H = \sum_{l,m} Q_{lm} \hat{z}_l \overline{\hat{z}_m} = \sum_{l,m} Q_{lm} f(v_l \cdot r) \overline{f(v_m \cdot r)}$$

Since our algorithm is randomized, we compute the expected objective value given solution $\hat{z}$. By linearity of expectation, we have:

$$\mathrm{E}[\hat{z}^H Q \hat{z}] = \sum_{l,m} Q_{lm} \mathrm{E}[f(v_l \cdot r) \overline{f(v_m \cdot r)}]$$

To that end, it would be sufficient to compute the quantity $\mathrm{E}[f(v_l \cdot r)\overline{f(v_m \cdot r)}]$ for any l, m, and this will be the main concern of our analysis. The analysis, of course, depends on the choice of the function $f(\cdot)$. However, the following Lemma will be useful and it is independent of the function $f(\cdot)$. Recall that for two vectors $b, c \in C^n$, we have $b \cdot c = \sum_{j=1}^n b_j \overline{c_j}$.

Lemma 1. *For any pair of vectors $b, c \in C^n$, $\mathrm{E}[(b \cdot r)\overline{(c \cdot r)}] = b \cdot c$, where $r = x + yi$ and $x \in R^n$ and $y \in R^n$ are two independent normally distributed random vector with mean 0 and co-variance matrix $\frac{1}{2}I$.*

Proof. This follows from a straightforward computation:

$$\mathrm{E}[(b \cdot r)\overline{(c \cdot r)}] = \mathrm{E}\left[\left(\sum_{j=1}^n b_j \overline{r}_j\right)\left(\sum_{k=1}^n \overline{c}_k r_k\right)\right] = \sum_{j,k=1}^n b_j \overline{c}_k \mathrm{E}[\overline{r}_j r_k] = \sum_{j=1}^n b_j \overline{c}_j$$

where the last equality follows from the fact that the entries of x and y are independent normally distributed with mean 0 and variance 1/2.

3 Discrete Problems Where Q Is Positive Semidefinite

In this section, we assume that Q is Hermitian and positive semidefinite. We consider the discrete complex quadratic optimization problem (3).

In this case, in the generic algorithm presented in Section 2, we specify the function $f(\cdot)$ as follows:

$$f(z) = \begin{cases} 1 & \text{if } \arg(z) \in [-\pi/k, \pi/k) \\ \omega & \text{if } \arg(z) \in [\pi/k, 3\pi/k) \\ \vdots & \vdots \\ \omega^{k-1} & \text{if } \arg(z) \in [(2k-3)\pi/k, (2k-1)\pi/k) \end{cases} \tag{6}$$

This way, we guarantee that $\hat{z}_j \in \{1, \omega, \ldots, \omega^{k-1}\}$ for $j = 1, 2, \cdots, n$, i.e. $\hat{z}$ is a feasible solution of problem (3).

Then, we can establish the following lemma:

Lemma 2.

$$\mathrm{E}[(b \cdot r)\overline{f(c \cdot r)}] = \frac{k \sin(\pi/k)}{2\sqrt{\pi}}(b \cdot c)$$

Proof. By rotation invariance, we may assume without loss of generality that $b = (b_1, b_2, 0, \ldots, 0)$ and $c = (1, 0, \ldots, 0)$. Then, we have:

$$\begin{aligned} \mathrm{E}[(b_1\overline{r_1} + b_2\overline{r_2})\overline{f(\overline{r_1})}] &= b_1 \mathrm{E}[\overline{r_1}\overline{f(\overline{r_1})}] \\ &= \frac{b_1}{\pi} \int_{\mathbf{R}} \int_{\mathbf{R}} (x - iy)\overline{f(x - iy)} \exp\{-(x^2 + y^2)\}\, dxdy \\ &= \frac{b_1}{\pi} \int_0^{\infty} \int_0^{2\pi} \rho^2 e^{-i\theta} \overline{f(\rho e^{-i\theta})} e^{-\rho^2}\, d\theta d\rho \end{aligned}$$

Now, observe that, for $j = 1, \ldots, k$, we have:

$$\int_{(2j-3)\pi/k}^{(2j-1)\pi/k} \overline{f(\rho e^{-i\theta})} e^{-i\theta}\, d\theta = \omega^{j-1} \int_{(2j-3)\pi/k}^{(2j-1)\pi/k} e^{-i\theta}\, d\theta = 2\sin(\pi/k)$$

In particular, the above quantity is independent of j. Moreover, since we have:

$$\int_0^{\infty} \rho^2 e^{-\rho^2}\, d\rho = \frac{\sqrt{\pi}}{4}$$

it follows that:

$$\mathrm{E}[(b_1\overline{r_1} + b_2\overline{r_2})\overline{f(\overline{r_1})}] = \frac{k \sin(\pi/k)}{2\sqrt{\pi}} b_1 = \frac{k \sin(\pi/k)}{2\sqrt{\pi}}(b \cdot c)$$

We are now ready to prove the main result of this section.

Theorem 1. *When Q is positive semidefinite and Hermitian, there exists an $\frac{(k\sin(\frac{\pi}{k}))^2}{4\pi}$–approximation algorithm for (3).*

Proof. It follows from Lemma 1 and Lemma 2 that:

$$\begin{aligned}
&\mathrm{E}\left[\left\{(b\cdot r)-\frac{2\sqrt{\pi}}{k\sin(\frac{\pi}{k})}f(b\cdot r)\right\}\overline{\left\{(c\cdot r)-\frac{2\sqrt{\pi}}{k\sin(\frac{\pi}{k})}f(c\cdot r)\right\}}\right]\\
&=-(b\cdot c)+\frac{4\pi}{(k\sin(\frac{\pi}{k}))^2}\mathrm{E}[f(b\cdot r)\overline{f(c\cdot r)}]
\end{aligned}$$

Therefore, we have:

$$\begin{aligned}
\mathrm{E}[\hat{z}^HQ\hat{z}] &= \frac{(k\sin(\frac{\pi}{k}))^2}{4\pi}\sum_{l=1}^n\sum_{m=1}^n q_{lm}(v_l\cdot v_m)\\
&\quad+\frac{(k\sin(\frac{\pi}{k}))^2}{4\pi}\sum_{l=1}^n\sum_{m=1}^n q_{lm}\mathrm{E}\left[\left\{(v_l\cdot r)-\frac{2\sqrt{\pi}}{k\sin(\frac{\pi}{k})}f(v_l\cdot r)\right\}\cdot\right.\\
&\qquad\left.\cdot\overline{\left\{(v_m\cdot r)-\frac{2\sqrt{\pi}}{k\sin(\frac{\pi}{k})}f(v_m\cdot r)\right\}}\right]\\
&\geq \frac{(k\sin(\frac{\pi}{k}))^2}{4\pi}\sum_{l=1}^n\sum_{m=1}^n q_{lm}(v_l\cdot v_m)=\frac{(k\sin(\frac{\pi}{k}))^2}{4\pi}\,w_{SDP} \qquad (7)
\end{aligned}$$

The last inequality is true since

$$\mathrm{E}\left[\left\{(v_l\cdot r)-\frac{2\sqrt{\pi}}{k\sin(\frac{\pi}{k})}f(v_l\cdot r)\right\}\overline{\left\{(v_m\cdot r)-\frac{2\sqrt{\pi}}{k\sin(\frac{\pi}{k})}f(v_m\cdot r)\right\}}\right]$$

is an inner product of two vectors in a Hilbert space, which together with the fact that Q is positive semidefinite shows that:

$$\sum_{l=1}^n\sum_{m=1}^n q_{lm}\mathrm{E}\left[\left\{(v_l\cdot r)-\frac{2}{\sqrt{\pi}}f(v_l\cdot r)\right\}\overline{\left\{(v_m\cdot r)-\frac{2}{\sqrt{\pi}}f(v_m\cdot r)\right\}}\right]\geq 0$$

It follows that our algorithm gives an $\frac{(k\sin(\frac{\pi}{k}))^2}{4\pi}$–approximation.

Now, let us consider problem (4) when Q is positive semidefinite. This problem can be seen as a special case of (3) by letting $k\to\infty$. In this case, the function $f(\cdot)$ defined in (6) is as follows:

$$f(t)=\begin{cases}\frac{t}{|t|} & \text{if } |t|>0\\ 0 & \text{if } t=0\end{cases} \qquad (8)$$

Note that as $k\to\infty$, we have $\frac{(k\sin(\frac{\pi}{k}))^2}{4\pi}\to\pi/4$. This establishes the following result, which has been proved independently by Ben–Tal, Nemirovski and Roos [2], and Zhang and Huang [12]. However, our proof is quite a bit simpler.

Corollary 1. *When Q is positive semidefinite and Hermitian, there exists an $\frac{\pi}{4}$–approximation algorithm for (4).*

Next, we show that our analysis is in fact tight for the continuous complex quadratic optimization problem (4). We give a family of examples which shows that the natural SDP relaxation for the above problem has an integrality gap arbitrarily close to $\pi/4$. We begin with a technical lemma.

Lemma 3. *Let u, v be two random, independent vectors on the unit sphere of C^p. Then, we have:*

$$\mathrm{E}[|u \cdot v|^2] = \frac{1}{p}; \qquad \mathrm{E}[|u \cdot v|] = \left(\frac{\sqrt{\pi}}{2} + o(1)\right)\frac{1}{\sqrt{p}}$$

Proof. Omitted in this extended abstract.

To construct the tight example, let p and $n \gg p$ be fixed. Let $v_1, \ldots, v_n$ be independent random vectors chosen uniformly according to the normalized Haar measure on the unit sphere of C^p. We define $A = (a_{ij})$ by $a_{ij} = \frac{1}{n^2}\overline{(v_i \cdot v_j)}$. By construction, the matrix A is positive semidefinite and Hermitian. Moreover, we have:

$$\sum_{i,j} a_{ij}(v_i \cdot v_j) = \frac{1}{n^2}\sum_{i,j} |v_i \cdot v_j|^2$$

By taking $n \to \infty$, the right–hand side converges to the average of the square of the inner product between two random vectors on the unit sphere of C^n. By Lemma 3, this value is $1/p$, and hence the optimal value of the SDP relaxation is at least $1/p$.

Now, let $z_i \in C$ be such that $|z_i| = 1$. Then, we have:

$$z^H A z = \sum_{i,j} a_{ij}\overline{z_i} z_j = \left|\frac{1}{n}\sum_{i=1}^{n} z_i v_i\right|^2$$

Hence, the value of the original SDP is the square of the maximum possible modulus of a vector $\frac{1}{n}\sum_{i=1}^{n} z_i v_i$. If we somehow know that the direction of this optimal vector is given by the unit vector c, then we must set $z_i = \overline{f(v_i \cdot c)}$ in order to maximize the modulus. It then follows that:

$$\frac{1}{n}\sum_{i=1}^{n} z_i v_i \cdot c = \left|\frac{1}{n}\sum_{i=1}^{n} z_i v_i\right|$$

by the Cauchy–Schwarz inequality. Moreover, this quantity converges to the average value of $|v \cdot c|$ as $n \to \infty$. By letting n arbitrarily large and choosing an appropriate ϵ–net of directions on the sphere, we conclude that with high probability, the value of the original SDP is at most $[(\sqrt{\pi}/2 + o(1))/\sqrt{p}]^2 = (\pi/4 + o(1))/p$, which yields the desired result.

4 Continuous Problems Where Q Is not Positive Semidefinite

In this section, we deal with problem (4) where the matrix Q is not positive semidefinite. However, for convenience, we assume that $w_{SDP} > 0$ such that the standard definition of approximation algorithm makes sense for our problem. It is clear that $w_{SDP} > 0$ as long as all the diagonal entries of Q are zeros.

Again, we use our generic algorithm presented in Section 2. In this case, we specify the function $f(\cdot)$ as follows:

$$f(t) = \begin{cases} \frac{t}{T} & \text{if } |t| \le T \\ \frac{t}{|t|} & \text{if } |t| > T \end{cases} \tag{9}$$

where T is a parameter which will be fixed later. If we let $z_j = f(v_j \cdot r)$, the solution $z = (z_1, \ldots, z_n)$ obtained by this rounding may not be feasible, as the point may not have unit modulus. However, we know that $|z_j| \le 1$. Thus, we can further round the solution as follows:

$$\hat{z} = \begin{cases} z/|z| & \text{with probability } (1 + |z|)/2 \\ -\bar{z}/|z| & \text{with probability } (1 - |z|)/2 \end{cases}$$

We then have the following:

Fact. For $i \neq j$, $\mathrm{E}[\hat{z}_i \overline{\hat{z}_j}] = \mathrm{E}[z_i \overline{z_j}]$.

This shows that the expected value of the solution on the circle equals that of the "fractional" solution obtained by applying $f(\cdot)$ to the SDP solution. Therefore, we could still restrict ourselves to the rounding function $f(\cdot)$.

Define

$$g(T) = \frac{1}{T} - \frac{1}{T} e^{-T^2} + \sqrt{\pi}(1 - \Phi(\sqrt{2}T))$$

where $\Phi(\cdot)$ is the probability distribution function of $N(0, 1)$.

Lemma 4. $\mathrm{E}[(b \cdot r)\overline{f(c \cdot r)}] = g(T)(b \cdot c)$

Proof. Again, without loss of generality, we assume that $c = (1, 0, \ldots, 0)$ and $b = (b_1, b_2, 0, \ldots, 0)$. Then, we have:

$$\begin{aligned} & \mathrm{E}[(b \cdot r)\overline{f(c \cdot r)}] \\ &= \mathrm{E}\left[(b_1 \bar{r}_1 + b_2 \bar{r}_2)\frac{r_1}{T}\Big||r_1| \le T\right] + \mathrm{E}\left[(b_1 \bar{r}_1 + b_2 \bar{r}_2)\frac{r_1}{|r_1|}\Big||r_1| > T\right] \\ &= \frac{1}{T}\mathrm{E}\left[b_1 |\bar{r}_1|^2 \Big||r_1| \le T\right] + \mathrm{E}\left[b_1 |\bar{r}_1|\Big||r_1| > T\right] \end{aligned}$$

$$
\begin{aligned}
&= \frac{b_1}{T} \cdot \frac{1}{\pi} \int_{x^2+y^2 \le T^2} (x^2+y^2)\ \exp(-(x^2+y^2)) dxdy \\
&\quad + \frac{b_1}{\pi} \int_{x^2+y^2 > T^2} \sqrt{x^2+y^2}\ \exp(-(x^2+y^2)) dxdy \\
&= \frac{b_1}{\pi T} \int_0^{2\pi} \int_0^T \rho^3 \exp(-\rho^2)\ d\rho d\theta + \frac{b_1}{\pi} \int_0^{2\pi} \int_T^{\infty} \rho^2 \exp(-\rho^2)\ d\rho d\theta \\
&= g(T) b_1
\end{aligned}
$$

where the last equality follows from the facts:

$$
\int_0^T \rho^3 \exp(-\rho^2) d\rho = \frac{1}{2} \left(1 - (T^2+1)\exp(-T^2)\right)
$$

and

$$
\int_T^{\infty} \rho^2 \exp(-\rho^2) d\rho = \frac{1}{2} \left(T\exp(-T^2) + \sqrt{\pi}(1 - \Phi(\sqrt{2}T)) \right)
$$

Lemma 5. $\mathrm{E}[f(c \cdot r)\overline{f(c \cdot r)}] = \frac{1}{T^2} - \frac{1}{T^2}\exp(-T^2)$

Proof. The proof is similar to that of Lemma 2. We again assume that $c = (1, 0, \ldots, 0)$.

$$
\begin{aligned}
&\mathrm{E}[f(c \cdot r)\overline{f(c \cdot r)}] \\
&= \mathrm{E}\left[\frac{\bar{r}_1}{T} \frac{r_1}{T} \middle| |r_1| \le T \right] + \mathrm{E}\left[\frac{\bar{r}_1}{|r_1|} \frac{r_1}{|r_1|} \middle| |r_1| > T \right] \\
&= \frac{1}{T^2} \cdot \frac{1}{\pi} \int_{x^2+y^2 \le T^2} (x^2+y^2) \exp(-(x^2+y^2))\ dxdy \\
&\quad + \frac{1}{\pi} \int_{x^2+y^2 > T^2} \exp(-(x^2+y^2))\ dxdy \\
&= \frac{1}{T^2} \cdot \frac{1}{\pi} \int_0^{2\pi} \int_0^T \rho^3 \exp(-\rho^2)\ d\rho d\theta + \frac{1}{\pi} \int_0^{2\pi} \int_T^{\infty} \rho \exp(-\rho^2)\ d\rho d\theta \\
&= \frac{1}{T^2}(1 - (T^2+1)\exp(-T^2)) + \exp(-T^2) \\
&= \frac{1}{T^2} - \frac{1}{T^2}\exp(-T^2)
\end{aligned}
$$

Theorem 2. *If* $T = 3\sqrt{\ln(n)}$, *then we have* $\mathrm{E}[\hat{z}^H Q \hat{z}] \ge \frac{1}{10\ln(n)}\ w_{SDP}$.

Proof. It follows from Lemma 1 and Lemma 4 that:

$$\mathrm{E}[\{(b\cdot r)-Tf(b\cdot r)\}\{\overline{(c\cdot r)-Tf(c\cdot r)}\}] = (1-2Tg(T))(b\cdot c)+T^2\mathrm{E}[f(b\cdot r)\overline{f(c\cdot r)}]$$

Then, we have:

$$\begin{aligned}\mathrm{E}[\hat{z}^H Q\hat{z}] = &\sum_{k=1}^{n}\sum_{m=1}^{n}\frac{2Tg(T)-1}{T^2}q_{km}(v_k\cdot v_m)\\ &+\frac{1}{T^2}\sum_{k=1}^{n}\sum_{m=1}^{n}q_{km}\mathrm{E}[\{(v_k\cdot r)-Tf(v_k\cdot r)\}\{\overline{(v_m\cdot r)-Tf(v_m\cdot r)}\}]\end{aligned}$$

Again, the quantity $\mathrm{E}[\{(b\cdot r)-Tf(b\cdot r)\}\{\overline{(c\cdot r)-Tf(c\cdot r)}\}]$ can be seen as an inner product of two vectors in a Hilbert space. Moreover, by letting $b=c$ and using Lemma 5, we know that the norm of an Euclidean unit vector in this Hilbert space is:

$$2-2Tg(T)-\exp(-T^2) = \exp(-T^2)-2T\sqrt{\pi}(1-\Phi(\sqrt{2}T))$$

It follows that:

$$\begin{aligned}&\frac{1}{T^2}\sum_{k=1}^{n}\sum_{m=1}^{n}q_{km}\mathrm{E}[\{(v_k\cdot r)-Tf(v_k\cdot r)\}\cdot\{\overline{(v_m\cdot r)-Tf(v_m\cdot r)}\}]\\ \geq &-\frac{\exp(-T^2)-2T\sqrt{\pi}(1-\Phi(\sqrt{2}T))}{T^2}\sum_{k=1}^{n}\sum_{m=1}^{n}|q_{km}|\end{aligned}$$

On the other hand, one can show that $w_{SDP}\geq\frac{1}{6n^3}\sum_{k,m}|q_{km}|>0$. Thus, we have:

$$\begin{aligned}&\frac{1}{T^2}\sum_{k=1}^{n}\sum_{m=1}^{n}q_{km}\mathrm{E}[\{(v_k\cdot r)-Tf(v_k\cdot r)\}\cdot\{\overline{(v_m\cdot r)-Tf(v_m\cdot r)}\}]\\ \geq &-\frac{\exp(-T^2)-2T\sqrt{\pi}(1-\Phi(\sqrt{2}T))}{T^2}\,6n^3\,w_{SDP}\end{aligned}$$

from which it follows that:

$$\begin{aligned}\mathrm{E}[\hat{z}^H Q\hat{z}] &\geq \left(\frac{2Tg(T)-1}{T^2}-\frac{\exp(-T^2)-2T\sqrt{\pi}(1-\Phi(\sqrt{2}T))}{T^2}6n^3\right)w_{SDP}\\ &\geq \frac{1-(2+6n^3)\exp(-T^2)}{T^2}\;w_{SDP}\end{aligned}$$

By letting $T=3\sqrt{\ln n}$, we have $\mathrm{E}[\hat{z}^H Q\hat{z}]\geq\frac{1}{10\ln n}w_{SDP}$ when $n\geq 3$ as desired.

References

1. Noga Alon, Assaf Naor, *Approximating the Cut–Norm via Grothendieck's Inequality*, Proc. 36th STOC, 2004.
2. Aharon Ben–Tal, Arkadi Nemirovski, Cornelis Roos, *Extended Matrix Cube Theorems with Applications to –Theory in Control*, Math. OR 28(3):497–523, 2003.
3. Moses Charikar, Anthony Wirth, *Maximizing Quadratic Programs: Extending Grothendieck's Inequality*, Proc. 45th FOCS, 2004.
4. Michel X. Goemans, Franz Rendl, *Combinatorial Optimization*, in *Handbook of Semidefinite Programming: Theory, Algorithms and Applications*, H. Wolkowicz, R. Saigal, L. Vandenberghe (eds.), Kluwer Academic Publishers, 2000.
5. Dorit S. Hochbaum (ed.), *Approximation Algorithms for NP–Hard Problems*, PWS Publishing Company, 1997.
6. Michel X. Goemans, David P. Williamson, *Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming*, Journal of the ACM 42(6):1115–1145, 1995.
7. Michel X. Goemans, David P. Williamson, *Approximation Algorithms for* Max–3–Cut *and Other Problems via Complex Semidefinite Programming*, JCSS 68(2):442–470, 2004.
8. Yurii Nesterov, *Global Quadratic Optimization via Conic Relaxation*, CORE Discussion Paper 9860, Université Catholique de Louvain, 1998.
9. Onur Toker, Hitay Özbay, *On the Complexity of Purely Complex Computation and Related Problems in Multidimensional Systems*, IEEE Transactions on Automatic Control 43(3):409–414, 1998.
10. Lieven Vandenberghe, Stephen Boyd, *Semidefinite Programming*, SIAM Review 38(1):49–95, 1996.
11. Yinyu Ye, *Approximating Quadratic Programming with Bound and Quadratic Constraints*, Math. Prog. 84:219–226, 1999.
12. Shuzhong Zhang, Yongwei Huang, *Complex Quadratic Optimization and Semidefinite Programming*, Technical Report SEEM2004–03, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, 2004.
13. Shuzhong Zhang, Yongwei Huang, *Private Communication*, 2004.

Semidefinite Bounds for the Stability Number of a Graph via Sums of Squares of Polynomials

Nebojša Gvozdenović* and Monique Laurent*

Centrum voor Wiskunde en Informatica, Kruislaan 413,
1098 SJ Amsterdam, The Netherlands

Abstract. Lovász and Schrijver [9] have constructed semidefinite relaxations for the stable set polytope of a graph $G = (V, E)$ by a sequence of lift-and-project operations; their procedure finds the stable set polytope in at most (G) steps, where (G) is the stability number of G. Two other hierarchies of semidefinite bounds for the stability number have been proposed by Lasserre [4],[5] and by de Klerk and Pasechnik [3], which are based on relaxing nonnegativity of a polynomial by requiring the existence of a sum of squares decomposition. The hierarchy of Lasserre is known to converge in (G) steps as it refines the hierarchy of Lovász and Schrijver, and de Klerk and Pasechnik conjecture that their hierarchy also finds the stability number after (G) steps. We prove this conjecture for graphs with stability number at most 8 and we show that the hierarchy of Lasserre refines the hierarchy of de Klerk and Pasechnik.

1 Introduction

Semidefinite programming plays an essential role for constructing good relaxations for hard combinatorial optimization problems, in particular, for the maximum stable set problem which will be considered in the present paper. Lovász [8] introduced the theta number $\vartheta(G)$ as an upper bound for the stability number $\alpha(G)$ of a graph G; $\vartheta(G)$ can be computed efficiently (to any arbitrary precision) using semidefinite programming and it coincides with $\alpha(G)$ when G is a perfect graph. Lovász and Schrijver [9] construct a hierarchy of semidefinite relaxations for the stable set polytope of G by a sequence of lift-and-project operations; their procedure is finite and it finds the stable set polytope in at most $\alpha(G)$ steps.

Two other hierarchies of semidefinite bounds for the stability number have been proposed by Lasserre [4, 5] and by de Klerk and Pasechnik [3]. They are based on the following paradigm: While testing nonnegativity of a polynomial is a hard problem, one can test efficiently whether a polynomial can be written as a sum of squares of polynomials via semidefinite programming. As was already proved by Hilbert in 1888 not every nonnegative multivariate polynomial can be written as a sum of squares (see Reznick [14] for a nice survey on this

* Supported by the Netherlands Organization for Scientific Research grant NWO 639.032.203.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 136–151, 2005.

topic). However, some representation theorems have been proved ensuring the existence of certain sums of squares decompositions under some assumption, like positivity of the polynomial on a compact basic closed semi-algebraic set (see, e.g., [18] for an exposition of such results). An early such result is due to Pólya [13] who showed that, if $p(x)$ is a homogeneous polynomial which is positive on $\mathbb{R}^n_+ \setminus \{0\}$, then $(\sum_{i=1}^n x_i)^r p(x)$ has only nonnegative coefficients (and thus $(\sum_{i=1}^n x_i^2)^r p(x_1^2, \ldots, x_n^2)$ is a sum of squares) for some sufficiently large integer r.

The starting point for Lasserre's construction is that the stability number $\alpha(G)$ of a graph $G = (V, E)$ can be expressed as the smallest scalar t for which the polynomial $t - \sum_{i \in V} x_i$ is nonnegative on the set $\{x \in \mathbb{R}^V \mid x_i x_j = 0 \ (ij \in E),\ x_i^2 = x_i \ (i \in V)\}$. Requiring the weaker condition that the polynomial $t - \sum_{i \in V} x_i$ can be written as a sum of squares modulo the ideal generated by $x_i x_j$ $(ij \in E)$ and $x_i^2 - x_i$ $(i \in V)$ with given degree bounds, yields a hierarchy of semidefinite upper bounds for $\alpha(G)$. The dual approach (in terms of moment matrices) yields the hierarchy of Lasserre [4, 5] of semidefinite relaxations for the stable set polytope. This hierarchy refines the hierarchy of Lovász and Schrijver (see [6]) and thus it also finds the stable set polytope in $\alpha(G)$ steps.

By a result of Motzkin and Straus [11], one may alternatively express $\alpha(G)$ as the smallest scalar t for which the matrix $M := t(I + A_G) - J$ (with entries $t - 1$ on the diagonal and at positions corresponding to edges and -1 elsewhere) is copositive, meaning that the polynomial $p_M(x) := \sum_{i,j \in V} x_i^2 x_j^2 M_{ij}$ is nonnegative on $\mathbb{R}^n$. Following Parrilo [12], de Klerk and Pasechnik [3] propose to relax the nonnegativity condition on $p_M(x)$ and to require instead that $(\sum_{i \in V} x_i^2)^r p_M(x)$ be a sum of squares for some integer $r \geq 0$. The convergence of these bounds to $\alpha(G)$ is guaranteed by the above mentioned result of Pólya. The first bound in the hierarchy coincides with the strengthening $\vartheta'(G)$ of the theta number introduced by McEliece, Rodemich and Rumsey [10] and Schrijver [16]. It is however not clear how the next bounds relate to the bounds provided by the construction of Lasserre. It is conjectured in [3] that the stability number is found after $\alpha(G)$ steps. In this paper we study this conjecture and develop a proof technique which enables us to show that the conjecture holds for graphs with stability number at most 8. Moreover, we show that the hierarchy of bounds of Lasserre (enhanced by adding some nonnegativity constraint) refines the hierarchy of bounds of de Klerk and Pasechnik, answering another open question of [3].

The paper is organized as follows. In Section 2, we first recall some definitions and results related to the hierarchies of bounds of Lasserre and of de Klerk and Pasechnik. Then we introduce a dual formulation for the latter bounds, which will enable us to compare the two hierarchies of bounds, and we present our main results. The proofs are delayed till Section 3, where we prove the conjecture for graphs with stability number at most 8, and till Section 4, where we prove the relation between the two hierarchies.

Throughout, $G = (V, E)$ denotes a graph with node set $V = \{1, \ldots, n\}$. Let $\alpha(G)$ denote its stability number, i.e., the largest cardinality of a stable set in G, and let A_G denote the adjacency matrix of G, i.e., A_G is the 0/1 matrix

indexed by V whose (i,j)-th entry is 1 when $i,j \in V$ are connected by an edge. All matrices are assumed to be symmetric and I, J, e, e_i $(i = 1,\ldots,n)$ denote, respectively, the identity matrix, the all-ones matrix, the all-ones vector, and the standard unit vectors of suitable sizes. A matrix M is *copositive* if $x^T Mx \geq 0$ for all $x \in \mathbb{R}^n_+$ and $\mathcal{C}_n$ denotes the *copositive cone*, consisting of the $n \times n$ copositive matrices. For a symmetric matrix M, we write $M \succeq 0$ if M is positive semidefinite. For a sequence $\beta \in \mathbb{Z}^n_+$, we set $|\beta| := \sum_{i=1}^n \beta_i$, $\beta! := \beta_1! \cdots \beta_n!$, $S(\beta) := \{i \mid \beta_i \neq 0\}$, and $S_{odd}(\beta) := \{i \mid \beta_i \text{ is odd}\}$. One says that β is even when $S_{odd}(\beta) = \emptyset$. Finally, set $I(n,r) := \{\beta \in \mathbb{Z}^n_+ \mid |\beta| = r\}$ and $\mathcal{P}_r(V) := \{I \subseteq V \mid |I| \leq r\}$.

2 Semidefinite Bounds for the Stability Number

2.1 The Semidefinite Bounds of Lasserre

Given an integer $r \geq 1$ and a vector $x = (x_I)_{I \in \mathcal{P}_{2r}(V)}$, consider the matrix:

$$M_r(x) := (x_{I \cup J})_{I,J \in \mathcal{P}_r(V)}$$

known as the moment matrix of x of order r. By setting:

$$las^{(r)}(G) := \max \textstyle\sum_{i \in V} x_i \;\text{ s.t. }\; \begin{array}{l} M_r(x) \succeq 0,\; x_I \geq 0 \;(I \subseteq V, |I| = r+1), \\ x_\emptyset = 1,\; x_{ij} = 0 \;(ij \in E) \end{array} \tag{1}$$

one obtains a hierarchy of semidefinite bounds for the stability number, known as Lasserre's hierarchy [5, 6]. Indeed, if S is a stable set, the vector $x \in \mathbb{R}^{\mathcal{P}_{2r}(V)}$ with $x_I = 1$ if $I \subseteq S$ and $x_I = 0$ otherwise, is feasible for (1) with objective value $|S|$, showing $\alpha(G) \leq las^{(r)}(G)$. We note that $las^{(1)}(G) = \vartheta'(G)$. For fixed r, the parameter $las^{(r)}(G)$ can be computed in polynomial time (to an arbitrary precision) since the semidefinite program (1) involves matrices of size $O(n^r)$ with $O(n^{2r})$ variables.

Equality $\alpha(G) = las^{(r)}(G)$ holds for $r \geq \alpha(G)$. This result remains valid if we remove the nonnegativity constraint: $x_I \geq 0$ $(|I| = r+1)$ in (1) ([6]). However, with this nonnegativity condition, we will be able to compare the hierarchies of Lasserre and de Klerk and Pasechnik (see Theorem 3 below).

2.2 The Semidefinite Bounds of de Klerk and Pasechnik

The starting point in [3] is the following formulation for $\alpha(G)$ found by Motzkin and Straus [11]:

$$\frac{1}{\alpha(G)} = \min \; x^T(I + A_G)x \;\text{ subject to } x \geq 0,\; \sum_{i=1}^n x_i = 1.$$

In other words,

$$\alpha(G) = \min \; t \;\text{ subject to } t(I + A_G) - J \in \mathcal{C}_n. \tag{2}$$

Therefore, upper bounds for $\alpha(G)$ can be obtained by replacing in program (2) the copositive cone $\mathcal{C}_n$ by a smaller subcone of it. Following [12], given an integer $r \geq 0$, $\mathcal{K}_n^{(r)}$ is the cone of $n \times n$ matrices M for which the polynomial

$$p_M^{(r)}(x) := \left(\sum_{i=1}^n x_i^2\right)^r \left(\sum_{i,j=1}^n M_{ij} x_i^2 x_j^2\right) \tag{3}$$

can be written as a sum of squares of polynomials. Parrilo [12] shows that

$$\mathcal{K}_n^{(0)} = \{P + N \mid P \succeq 0, N \geq 0\}. \tag{4}$$

A characterization of $\mathcal{K}_n^{(1)}$ can be found in [12, 1]. Obviously, $\mathcal{K}_n^{(r)} \subseteq \mathcal{K}_n^{(r+1)} \subseteq \ldots \subseteq \mathcal{C}_n$. The result of Pólya mentioned in the Introduction shows that the interior of the cone $\mathcal{C}_n$ is equal to $\bigcup_{r\geq 0} \mathcal{K}_n^{(r)}$. Setting

$$\vartheta^{(r)}(G) := \min\ t \ \text{ subject to } \ t(I + A_G) - J \in \mathcal{K}_n^{(r)}, \tag{5}$$

one obtains a hierarchy of upper bounds for $\alpha(G)$. The first bound $\vartheta^{(0)}(G)$ is equal to

$$\vartheta'(G) = \max\ \mathrm{Tr}(JX) \ \text{ s.t. } \mathrm{Tr}(X) = 1,\ X_{ij} = 0\ (ij \in E),\ X \succeq 0,\ X \geq 0 \tag{6}$$

(see [3]). Thus, $\vartheta^{(0)}(G) \leq \vartheta(G)$, since program (6) without the nonnegativity condition is a formulation of the theta number.

The problem of finding a sum of squares decomposition for a polynomial of degree $2d$ can be formulated as a semidefinite program involving matrices of size $O(n^d)$ and $O(n^{2d})$ variables (see, e.g., [12]). Therefore, for fixed r, program (5) can be reformulated as a semidefinite program of polynomial size and thus $\vartheta^{(r)}(G)$ can be computed in polynomial time (to any precision).

De Klerk and Pasechnik [3] show that

$$\alpha(G) = \lfloor \vartheta^{(r)}(G) \rfloor \text{ for } r \geq \alpha(G)^2.$$

Indeed the matrix $M := \alpha(1+\epsilon)(I + A_G) - J$ with $\alpha = \alpha(G)$ and $\epsilon = \frac{\alpha-1}{\alpha^2-\alpha+1}$, belongs to the cone $\mathcal{K}_n^{(r)}$ since all the coefficients of the polynomial $p_M^{(r)}(x)$ are nonnegative; this implies that $\alpha(G) \leq \vartheta^{(r)}(G) \leq \alpha(G)(1+\epsilon) < \alpha(G) + 1$.

Let us observe that, for the matrix $M := \alpha(I + A_G) - J$, the polynomial $p_M^{(r)}(x)$ has a negative coefficient for any $r \geq 0$ when $\alpha = \alpha(G) \geq 2$. To see it, recall from [1] that

$$p_M^{(r)}(x) = \sum_{\beta \in I(n,r+2)} \frac{r!}{\beta!} c_\beta x^{2\beta}, \text{ where } c_\beta := \beta^T M \beta - \beta^T \mathrm{diag}(M). \tag{7}$$

If $S(\beta)$ is a stable set, then $c_\beta = \alpha \sum_i \beta_i(\beta_i - 1) - (r+1)(r+2)$. Write $r+2 = q\alpha + s$ with $q, s \in \mathbb{Z}_+$, $0 \leq s < \alpha$; then $c_\beta < 0$ for $\beta = (q+1, \ldots, q+1, q, \ldots, q, 0, \ldots 0)$

with s entries equal to $q+1$, $\alpha - s$ entries equal to q, and $S(\beta)$ being a stable set.

It is also shown in [3] that

$$\vartheta^{(1)}(G) \leq 1 + \max_{i \in V} \vartheta^{(0)}(G \backslash i^{\perp}) \tag{8}$$

where, for $i \in V$, $G \backslash i^{\perp}$ is the graph obtained from G by deleting i and its neighbours. Therefore, $\vartheta^{(1)}(G) = \alpha(G)$ when $\alpha(G) \leq 2$. More generally, de Klerk and Pasechnik [3] conjecture:

Conjecture 1. $\vartheta^{(r)}(G) = \alpha(G)$ for $r \geq \alpha(G) - 1$.

2.3 Dual Formulation

Using conic duality, the bound $\vartheta^{(r)}(G)$ from (5) can be reformulated as

$$\vartheta^{(r)}(G) = \max \ \mathrm{Tr}(JX) \ \text{ subject to } \mathrm{Tr}((I + A_G)X) = 1, \ X \in (\mathcal{K}_n^{(r)})^*. \tag{9}$$

As the programs (5) and (9) are strictly feasible, there is no duality gap and the optima in (5) and (9) are indeed attained ([3]). For $r = 0$, it follows from (4) that $(\mathcal{K}_n^{(0)})^*$ is the cone of completely positive (i.e., positive semidefinite and nonnegative) matrices. For $r \geq 1$, one can give an explicit description of the dual cone $(\mathcal{K}_n^{(r)})^*$ in terms of moment matrices.

Definition 1. *Let $y = (y_\delta)_{\delta \in I(n,2r+4)}$ be given.*

(i) Define the matrix $N_{r+2}(y)$ indexed by $I(n, r+2)$, whose (β, β')-th entry is equal to $y_{\beta+\beta'}$, for $\beta, \beta' \in I(n, r+2)$.

(ii) For $\gamma \in I(n,r)$, $N^\gamma(y)$ denotes the principal submatrix of $N_{r+2}(y)$ indexed by $\gamma + 2e_1, \ldots, \gamma + 2e_n$; that is, $N^\gamma(y)$ is the $n \times n$ matrix with (i,j)-th entry $y_{2\gamma+2e_i+2e_j}$, for $i, j = 1, \ldots, n$.

(iii) Define the $n \times n$ matrix

$$C(y) := \sum_{\gamma \in I(n,r)} \frac{r!}{\gamma!} N^\gamma(y). \tag{10}$$

Definition 2. *Define the cone*

$$\mathcal{C}_n^{(r)} := \{Z \in \mathbb{R}^{n \times n} \mid Z = C(y) \ \text{ for some } y \in \mathbb{R}^{I(n,2r+4)} \text{ with } N_{r+2}(y) \succeq 0\}.$$

As the matrix $C(y)$ in (10) involves only entries of y indexed by even sequences, one can assume w.l.o.g. in the definition of the cone $\mathcal{C}_n^{(r)}$ that $y_\delta = 0$ whenever δ has an odd component.

Lemma 1. *The cones $\mathcal{K}_n^{(r)}$ and $\mathcal{C}_n^{(r)}$ are dual of each other; i.e., $\mathcal{C}_n^{(r)} = (\mathcal{K}_n^{(r)})^*$ and $\mathcal{K}_n^{(r)} = (\mathcal{C}_n^{(r)})^*$.*

Proof. For a polynomial $p(x) = \sum_\delta p_\delta x^\delta$, let $p := (p_\delta)_\delta$ denote the vector of its coefficients. The following can be easily verified:

$$y^T v = u^T(N_{r+2}(y))u \quad \text{for } y \in \mathbb{R}^{I(n,2r+4)},\ u \in \mathbb{R}^{I(n,r+2)},\ v(x) := u(x)^2. \tag{11}$$

Consider the cones $\mathcal{C} := \{y \in \mathbb{R}^{I(n,2r+4)} \mid N_{r+2}(y) \succeq 0\}$, $\mathcal{D} := \{p \in \mathbb{R}^{I(n,2r+4)} \mid$ the polynomial $p(x)$ is a sum of squares$\}$. Then, $\mathcal{C} = \mathcal{D}^*$ as a direct application of (11), which implies $\mathcal{D} = \mathcal{C}^*$ since $\mathcal{D}$ is a closed cone (see [15]). Using (7), one can also easily verify that

$$Tr(MC(y)) = y^T(p_M^{(r)}) \quad \text{for } y \in \mathbb{R}^{I(n,2r+4)},\ M \text{ symmetric } n \times n \text{ matrix} \tag{12}$$

where $p_M^{(r)}(x)$ is the polynomial from (3). We can now prove the lemma. As $\mathcal{C}_n^{(r)}$ is a closed cone, it suffices to show: $\mathcal{K}_n^{(r)} = (\mathcal{C}_n^{(r)})^*$. The inclusion $\mathcal{K}_n^{(r)} \subseteq (\mathcal{C}_n^{(r)})^*$ follows using (11) and (12). Conversely, let $M \in (\mathcal{C}_n^{(r)})^*$. Then, by (12), $y^T(p_M^{(r)}) \geq 0$ for all $y \in \mathcal{C}$; that is, $p_M^{(r)} \in \mathcal{C}^* = \mathcal{D}$, showing that $M \in \mathcal{K}_n^{(r)}$. □

Consider the program

$$\begin{aligned} \tilde{\vartheta}^{(r)}(G) := \max\ \mathrm{Tr}(X)\ \ \text{s.t.}\ \ & X \in \mathcal{C}_n^{(r)} = (\mathcal{K}_n^{(r)})^*,\ \mathrm{Tr}(A_G X) = 0, \\ & X - \mathrm{diag}(X)\mathrm{diag}(X)^T \succeq 0. \end{aligned} \tag{13}$$

Then,

$$\alpha(G) \leq \tilde{\vartheta}^{(r)}(G) \leq \vartheta^{(r)}(G). \tag{14}$$

Indeed, if X is feasible for (13), then $X' := \frac{X}{Tr(X)}$ is feasible for (9) with $Tr(JX') \geq Tr(X)$, which shows $\tilde{\vartheta}^{(r)}(G) \leq \vartheta^{(r)}(G)$. Given a stable set S with incidence vector $x := \chi^S$, define the vector $y \in \mathbb{R}^{I(n,2r+4)}$ with $y_\delta = \frac{1}{|S|^r}$ if δ is even and $S(\delta) \subseteq S$, and $y_\delta = 0$ otherwise. Then, $N_{r+2}(y) \succeq 0$; $X := C(y) = xx^T$ is feasible for (13) with $Tr(X) = |S|$, which shows $\alpha(G) \leq \tilde{\vartheta}^{(r)}(G)$.

2.4 The Main Results

Our main results are the following:

Theorem 1. *For a graph G and a positive integer $r \leq \min(\alpha(G) - 1, 6)$,*

$$\vartheta^{(r)}(G) \leq r + \max_{S \subseteq V \text{ stable},\ |S|=r} \vartheta^{(0)}(G \backslash S^\perp), \tag{15}$$

where $S^\perp$ denotes the set of nodes that belong to S or are adjacent to a node in S.

Theorem 2. *Conjecture 1 holds for $\alpha(G) \leq 8$; that is,*

$$\vartheta^{(\alpha(G)-1)}(G) = \alpha(G) \quad \text{if } \alpha(G) \leq 8.$$

Theorem 3. *For $r \geq 1$, the parameters from (1),(9) and (13) satisfy:*

$$las^{(r)}(G) \leq \tilde{\vartheta}^{(r-1)}(G) \leq \vartheta^{(r-1)}(G). \tag{16}$$

Theorem 2 follows directly from Theorem 1 when $\alpha(G) \leq 7$ and from its proof technique when $\alpha(G) = 8$. Our proof technique does not apply to the case when $\alpha(G) \geq 9$. It is quite more complicated than the proof of convergence in $\alpha(G)$ steps for the Lovász-Schrijver and the Lasserre semidefinite hierarchies. One of the main difficulties (as pointed out later in the proof) comes from the fact that, for $r \geq 1$, the cone $\mathcal{K}_n^{(r)}$ is not invariant under some simple matrix operations, like extending a matrix by adding a zero row and column to it, or rescaling it by positive multipliers (which obviously preserve copositivity and positive semidefiniteness). For instance, when G is a circuit of length 5, the matrix $M := 2(I + A_G) - J$ belongs to $\mathcal{K}_5^{(1)}$, but adding a zero row and column yields a matrix that does not belong to $\mathcal{K}_6^{(1)}$. We thank E. de Klerk for communicating this example to us.

As Theorem 3 shows, the bound $las^{(r)}(G)$ is at least as good as $\tilde{\vartheta}^{(r-1)}(G)$. There exist in fact graphs for which strict inequality: $las^{(2)}(G) < \tilde{\vartheta}^{(1)}(G)$ holds. For this, given integers $2 \leq d \leq n$, consider the graph $G(n,d)$ with node set $\mathcal{P}(V)$ ($|V| = n$) where $I, J \in \mathcal{P}(V)$ are connected by an edge if $|I \Delta J| \in \{1, \ldots, d-1\}$. Then $\alpha(G(n,d))$ is the maximum cardinality of a binary code of word length n with minimum distance d. Delsarte [2] introduced a linear programming bound which coincides with the parameter $\vartheta'(G(n,d))$ ([16]). Schrijver [17] introduced a stronger semidefinite bound which roughly[1] lies between the bounds $las^{(1)}(G(n,d))$ and $las^{(2)}(G(n,d))$ ([7]). While $G(n,d)$ has 2^n vertices, Schrijver's bound can be computed via a semidefinite program of size $O(n^3)$ (using a block-diagonalization of the underlying Terwiliger algebra). It turns out that the same algebraic property holds for the bound $\vartheta^{(1)}(G(n,d))$; thus we could compute this bound as well as Schrijver's bound for the parameters $(n,d) = (17,4), (17,6), (17,8)$, and we found:

$$las^{(2)}(G(17,4)) \leq 3276 < 3607 \leq \vartheta^{(1)}(G(17,4))$$
$$las^{(2)}(G(17,6)) \leq 352 < 395 \leq \vartheta^{(1)}(G(17,6))$$
$$las^{(2)}(G(17,8)) \leq 41 < 42 \leq \vartheta^{(1)}(G(17,8)).$$

3 Proofs of Theorems 1 and 2

Let $G = (V,E)$ be a graph with stability number $\alpha(G)$, $V = \{1, \ldots, n\}$ and $1 \leq r \leq \alpha(G) - 1$ an integer. Set

$$t := r + \max_{S \subseteq V stable, |S| = r} \vartheta^{(0)}(G \backslash S^{\perp}).$$

Then, $t \geq r + 1$. By assumption,

$$(t-r)(I + A_{G \backslash S^{\perp}}) - J \in \mathcal{K}^{(0)}_{n-|S^{\perp}|} \text{ for any stable set } S \text{ in } G \text{ of size } r. \quad (17)$$

[1] Indeed, the formulation of Schrijver's bound has an additional constraint, namely, $_{ijk} \leq _{ij}$ for all $i,j,k \in V$, which does not appear in the definition of the bound $la^{(r)}(G)$ used in the present paper.

In order to prove Theorem 1, we have to show that, for $1 \leq r \leq \min(\alpha(G)-1, 6)$,

$$M := t(I + A_G) - J \in \mathcal{K}_n^{(r)}. \tag{18}$$

We need some notation. For two nodes $u, v \in V$, write $u \simeq v$ if $u = v$ or $uv \in E$, and $u \not\simeq v$ otherwise. For $x \in \mathbb{R}^n$, set $v(x) := (x_i^2)_{i=1}^n$. Let B be an $m \times n$ matrix. We say that B is a $q \times s$ *block matrix* if the set $\{1, \ldots, m\}$ indexing its rows can be partitioned into $Q_1 \cup \ldots \cup Q_q$ and the set $\{1, \ldots, n\}$ indexing its columns can be partitioned into $S_1 \cup \ldots \cup S_s$ in such a way that, for any $h \in \{1, \ldots, q\}$, $h' \in \{1, \ldots, s\}$, the entries B_{ij} for $i \in Q_h, j \in S_{h'}$ are all equal to the same value, say $\tilde{b}_{hh'}$. In other words, B is obtained from the matrix $\tilde{B} := (\tilde{b}_{hh'})_{\substack{h \in \{1,\ldots,q\} \\ h' \in \{1,\ldots,s\}}}$ by suitably duplicating rows and columns. Obviously, $B \succeq 0$ if and only if $\tilde{B} \succeq 0$. We call $\tilde{B}$ the *skeleton* of the block matrix B.

The following observation plays a central role in the proof.

Lemma 2. *Let $X(i)$ $(i \in V)$ be symmetric matrices satisfying the condition:*

$$X(i)_{jk} + X(j)_{ik} + X(k)_{ij} \geq 0 \quad \textit{for all } i, j, k \in V, \tag{19}$$

then the polynomial $\sum_{i \in V} x_i^2 v(x)^T X(i) v(x) = \sum_{i,j,k \in V} x_i^2 x_j^2 x_k^2 X(i)_{jk}$ is a sum of squares.

Proof. The polynomial $\sum_{i,j,k \in V} x_i^2 x_j^2 x_k^2 X(i)_{jk}$ is equal to

$$\begin{aligned} &\sum_{\substack{(i,j,k) \in V^3 \\ i \neq j \neq k \neq i}} x_i^2 x_j^2 x_k^2 [X(i)_{jk} + X(i)_{jk} + X(i)_{jk}] \\ &+ \sum_{\substack{(i,j) \in V^2 \\ i \neq j}} x_i^2 x_j^4 [X(i)_{jj} + 2X(j)_{ij}] + \sum_{i \in V} x_i^6 X(i)_{ii}, \end{aligned}$$

which is a sum of squares, since all coefficients are nonnegative by (19). □

Our strategy will be to construct matrices $X(\{i_1, ..., i_k\}, i)$ $(i \in V)$ satisfying (19) when $\{i_1, ..., i_k\}$ is a stable set of size $k \leq r$. We will use them to recursively decompose M into $M - X(i_1) - X(i_1, i_2) - \ldots - X(i_1, \ldots, i_k)$ in such way that at the last level $k = r$ we obtain matrices in $\mathcal{K}_n^{(0)}$.

3.1 Defining Sets of Matrices Satisfying the Linear Condition (19)

Let S be a stable set of cardinality k, $0 \leq k \leq r$. We define a set of matrices $X(S, i)$ (for $i \in V$) indexed by V that satisfy the condition (19). Set $m_0 := 1$ and $m_k := \frac{t^k}{(t-1)\cdots(t-k)}$ for $k = 1, \ldots, r$. (Then, $t \geq r + 1 > k$.)

For $i \in S^\perp$, $X(S, i)$ is the symmetric matrix whose entry at position (u, v) is defined as follows:

$$m_k \text{ times } \begin{cases} 0 & \text{if } u \text{ or } v \in S^\perp \\ t - k - 1 & \text{if } u, v \in V \setminus S^\perp \text{ and } u \simeq v \\ -1 & \text{if } u, v \in V \setminus S^\perp \text{ and } u \not\simeq v. \end{cases}$$

For $i \notin S^\perp$, $X(S,i)$ is the symmetric matrix whose entry at position (u,v) is defined as follows:

$$m_k \text{ times } \begin{cases} 0 & \text{if } u,v \in S^\perp \\ -\frac{t-k-1}{2} & \text{if } u \in S^\perp, v \in i^\perp \setminus S^\perp \\ \frac{1}{2} & \text{if } u \in S^\perp,\ v \in V \setminus (S^\perp \cup i^\perp) \\ 0 & \text{if } u,v \in i^\perp \setminus S^\perp \text{ and } u \simeq v \\ -(t-k) & \text{if } u,v \in i^\perp \setminus S^\perp \text{ and } u \not\simeq v \\ t-\frac{k}{2} & \text{if } u \in i^\perp \setminus S^\perp,\ v \in V \setminus (S^\perp \cup i^\perp) \text{ and } u \simeq v \\ \frac{k}{2} & \text{if } u \in i^\perp \setminus S^\perp,\ v \in V \setminus (S^\perp \cup i^\perp) \text{ and } u \not\simeq v \\ -k & \text{if } u,v \in V \setminus (S^\perp \cup i^\perp) \text{ and } u \simeq v \\ 0 & \text{if } u,v \in V \setminus (S^\perp \cup i^\perp) \text{ and } u \not\simeq v. \end{cases}$$

If $S = \{i_1, \ldots, i_k\}$, we also denote $X(S,i)$ as $X(i_1, \ldots, i_k, i)$. When $S = \emptyset$, we set $X(\emptyset, i) =: X(i)$. Given an ordering $(S) = (i_1, \ldots, i_k)$ of the elements of S, define the matrix

$$M((S)) := M - X(i_1) - X(i_1, i_2) - \ldots - X(i_1, \ldots, i_k). \tag{20}$$

Lemma 3. *Given a stable set S of size $0 \le k \le r$, the matrices $X(S,i)$ $(i \in V)$ satisfy (19).*

Proof. Direct verification. □

3.2 The Role of the Matrices $X(S,i)$ and $M((S))$ in the Proof

Our objective is to prove that the matrix M from (18) belongs to the cone $\mathcal{K}_n^{(r)}$, i.e., that the polynomial $p_M^{(r)}(x) = \sigma(x)^r v(x)^T M v(x)$ is a sum of squares, setting $\sigma(x) := \sum_{i=1}^n x_i^2$. The basic idea is to decompose $p_M^{(r)}(x)$ as

$$\sigma(x)^{r-1} \sum_{i=1}^n x_i^2 v(x)^T (M - X(i)) v(x) + \sigma(x)^{r-1} \sum_{i=1}^n x_i^2 v(x)^T X(i) v(x). \tag{21}$$

The second sum is a sum of squares by Lemmas 2 and 3. Each matrix $M - X(i)$ can be written as

$$M - X(i) = \begin{array}{c} \\ i^\perp \\ V \setminus i^\perp \end{array} \begin{array}{c} \begin{array}{cc} i^\perp & V \setminus i^\perp \end{array} \\ \begin{pmatrix} (t-1)J & -J \\ -J & t(I + A_{G \setminus i^\perp}) - J \end{pmatrix} \end{array} \tag{22}$$

$$= \frac{t}{t-1} \begin{pmatrix} 0 & 0 \\ 0 & (t-1)(I + A_{G \setminus i^\perp}) - J \end{pmatrix} + \begin{pmatrix} (t-1)J & -J \\ -J & \frac{1}{t-1} J \end{pmatrix}. \tag{23}$$

When $r = 1$, (22),(23) together with assumption (17) imply that $M - X(i) \in \mathcal{K}_n^{(0)}$ and thus $p_M^{(1)}(x)$ is a sum of squares; therefore, (8) holds. Assume now $r \ge 2$. The last matrix in (23) is positive semidefinite. Suppose our assumption would

be that $(t-1)(I - A_{G\setminus i^\perp}) - J \in \mathcal{K}^{(r-1)}_{n-|i^\perp|}$, then it would be tempting to conclude from (22) and (23) that $M - X(i) \in \mathcal{K}^{(r-1)}_n$ (which would then imply that $M \in \mathcal{K}^{(r)}_n$ and thus conclude the proof). This would be correct if we would work with cones of matrices which are closed under adding a zero row and column, but this is not the case for the cones $\mathcal{K}^{(r)}$ and thus this argument does not work. To go around this difficulty, we further decompose the first sum in (21) by developing $\sigma(x)^{r-1}$ as $\sigma(x)^{r-2}\sum_{j=1}^n x_j^2$ and using the matrices $X(i,j)$. Generally, one can write the following 'inclusion-exclusion' formula for the matrix $\sigma(x)^r M$:

$$\begin{aligned}\sigma(x)^r M = &\sum_{h=1}^{r} \sigma(x)^{r-h} \sum_{\substack{i_1\in V,\ i_2\notin i_1^\perp,\dots,i_{h-1}\notin i_1^\perp\cup\dots\cup i_{h-2}^\perp \\ i_h\in V}} x_{i_1}^2\cdots x_{i_h}^2 X(i_1,\dots,i_h)\\ &+\sum_{h=2}^{r} \sigma(x)^{r-h} \sum_{\substack{i_1\in V,\ i_2\notin i_1^\perp,\dots,i_{h-1}\notin i_1^\perp\cup\dots\cup i_{h-2}^\perp \\ i_h\in i_1^\perp\cup\dots\cup i_{h-1}^\perp}} x_{i_1}^2\cdots x_{i_h}^2 M((i_1,\dots,i_h))\\ &+\sum_{\substack{i_1\in V,\ i_2\notin i_1^\perp,\dots,i_{r-1}\notin i_1^\perp\cup\dots\cup i_{r-2}^\perp \\ i_r\notin i_1^\perp\cup\dots\cup i_{r-1}^\perp}} x_{i_1}^2\cdots x_{i_r}^2 M((i_1,\dots,i_r)).\end{aligned} \tag{24}$$

Therefore, in order to show that $M \in \mathcal{K}^{(r)}_n$, it suffices to show that

$$\begin{gathered}M((i_1,\dots,i_k,i_{k+1})) \in \mathcal{K}^{(0)}_n \text{ for } S := \{i_1,\dots,i_k\} \text{ stable,}\\ i_{k+1}\in S^\perp,\ 1\le k\le r-1,\end{gathered} \tag{25}$$

and

$$M((i_1,\dots,i_r)) \in \mathcal{K}^{(0)}_n \text{ for } \{i_1,\dots,i_r\} \text{ stable.} \tag{26}$$

For this we need to study the structure of the matrices $M((S))$.

3.3 The Structure of the Matrices $M((S))$

Given an ordered stable set $(S) = (i_1, i_2, ..., i_k)$ with $k = 1,\dots,r$, consider the matrix $M((S))$ from (20) and write

$$M((S)) := \begin{array}{c} \\ S^\perp \\ V\setminus S^\perp \end{array}\!\!\begin{array}{c} \begin{array}{cc} S^\perp & V\setminus S^\perp \end{array} \\ \begin{pmatrix} C_k(S) & D_k(S) \\ D_k(S)^T & E_k(S) \end{pmatrix} \end{array}. \tag{27}$$

Lemma 4. *The matrix $M((S))$ from (27) has the following properties.*

(i) $C_k(S)$ is a $k\times k$ block matrix whose rows and columns are indexed by the partition of $S^\perp$ into $i_1^\perp \cup (i_2^\perp \setminus i_1^\perp) \cup \dots \cup (i_k^\perp \setminus \{i_1,\dots,i_{k-1}\}^\perp)$. Let C_k be the skeleton of $C_k(S)$ (C_k is a $k\times k$ matrix) and set $d_k := C_k e \in \mathbb{R}^k$. Then,

$$e^T C_k e = \sum_{h=1}^{k} d_k(h) = (m_k - 1)(t-k)^2.$$

(ii) *The matrix $D_k(S)$ is a $k \times 1$ block matrix, with the same partition as above for the set $S^\perp$ indexing its rows. Given $h \in \{1, \ldots, k\}$, all entries in the $(h,1)$-block take the same value, which is equal to $-\frac{d_k(h)}{t-k}$.*

(iii) *For $u, v \in V \setminus S^\perp$, the (u,v)-th entry of $E_k(S)$ is equal to $tm_{k-1}-1$ if $u \simeq v$ and to -1 if $u \not\simeq v$.*

Proof. The block structure of the matrices C_k and D_k is determined by the construction of the matrix $M((S))$ in (20) and the shape of the matrices $X(.)$ defined in Section 3.1. We show the lemma by induction on $k \geq 1$. For $k = 1$, the matrix $M((S)) = M - X(i_1)$ has the shape given in (22) and the desired properties hold. Assume (i),(ii),(iii) hold for a stable set S of size $k \geq 1$. Let $i \in V \setminus S^\perp$. We show that (i),(ii),(iii) hold for the stable set $S \cup \{i\}$. Let $D'_k(S)$ (resp., $D''_k(S)$) be the submatrices of $D_k(S)$ whose columns are indexed by $i^\perp \setminus S$ (resp., $V \setminus (S \cup i^\perp)$) and with the same row indices as $D_k(S)$. Then $C_{k+1}(S,i)$ and $D_{k+1}(S,i)$ have the following block structure:

$$C_{k+1}(S,i) = \begin{pmatrix} C_k(S) & D'_k(S) + \frac{t-k-1}{2} m_k J \\ D'_k(S)^T + \frac{t-k-1}{2} m_k J^T & (tm_{k-1}-1)J \end{pmatrix} \tag{28}$$

$$D_{k+1}(S,i) = \begin{pmatrix} D''_k(S) - \frac{1}{2} m_k J \\ (-1 - m_k \frac{k}{2})J \end{pmatrix}, \tag{29}$$

where J denotes the all-ones matrix of appropriate size. By simple calculation one can show that $C_{k+1}(S,i)$ and $D_{k+1}(S,i)$ satisfy the induction hypothesis.

Finally, the (u,v)-th entry of the matrix $E_{k+1}(S,i)$ remains the same as in $E_k(S)$, i.e., equal to -1, if $u \not\simeq v$ and, for $u \simeq v$, it is equal to $tm_{k-1} - 1 + km_k = (t-k)m_k - 1 + km_k = tm_k - 1$. □

Corollary 1. *Let S be a stable set of size $k = 1, \ldots, r$. Then,*

$$G((S)) := \begin{pmatrix} C_k(S) & D_k(S) \\ D_k(S)^T & (m_k - 1)J \end{pmatrix} \succeq 0 \Longleftrightarrow C_k(S) \succeq 0, \tag{30}$$

$$M((S)) = G((S)) + m_k \begin{pmatrix} 0 & 0 \\ 0 & (t-k)(I + A_{G \setminus S^\perp}) - J \end{pmatrix}, \tag{31}$$

$$M((S,i)) = G((S)) \text{ if } i \in S^\perp. \tag{32}$$

Proof. By Lemma 4, $C_k(S)$, $D_k(S)$ are block matrices; hence $G((S)) \succeq 0$ if and only if its skeleton $G := \begin{pmatrix} C_k & -\frac{1}{t-k} C_k e \\ -\frac{1}{t-k} e^T C_k & m_k - 1 \end{pmatrix}$ is positive semidefinite. Now, $G \succeq 0 \Longleftrightarrow C_k \succeq 0$ since the last column of G is a linear combination of the first k columns; thus (30) holds. Relations (31), (32) follow using the definitions. □

Therefore, (25), (26) hold (and thus $M \in \mathcal{K}_n^{(r)}$) if we can show that $C_k(S) \succeq 0$ for any stable set S of size $k \leq r$. As $C_k(S)$ is a block matrix, it suffices to show that its skeleton C_k is positive semidefinite. Moreover, it suffices to show that $C_r \succeq 0$ since, in view of (28), the matrices C_k $(1 \leq k \leq r)$ are in fact the leading principal submatrices of C_r.

3.4 The Matrix C_r Is Positive Semidefinite When $r \leq \min(\alpha(G) - 1, 6)$

Recall that the entries of C_r depend on the parameter t; thus one may alternatively write C_r as $C_r(t)$. Our task is now to show that $C_r(t) \succeq 0$ for all $t \geq r+1$ and $r \leq \min(\alpha(G) - 1, 6)$. We achieve this by proving that

$$\det C_k(t) > 0 \text{ for } t \geq r+1,\ k = 1, \ldots, r. \tag{33}$$

The proof for (33) relies on establishing a recurrence relationship among the determinants of $C_k(t)$. We need the following lemma.

Lemma 5. *Assume C_{k+1} is nonsingular. Then,*

$$e^T (C_{k+1})^{-1} e = \frac{t^2}{(t-k)^2} \frac{\det C_k}{\det C_{k+1}}. \tag{34}$$

Proof. Write $C_{k+1} := \begin{pmatrix} C_k & x \\ x^T & a \end{pmatrix}$, $(C_{k+1})^{-1} := \begin{pmatrix} A & y \\ y^T & b \end{pmatrix}$. Then,

$$\text{(a) } AC_k + yx^T = I;\ \text{(b) } C_k y + bx = 0;\ \text{(c) } Ax + ay = 0;\ \text{(d) } x^T y + ab = 1. \tag{35}$$

By Lemma 4 and (28), $a = tm_{k-1} - 1 = (t-k)m_k - 1$ and $x = \rho_k e - \frac{1}{t-k} C_k e$, setting $\rho_k := m_k \frac{t-k-1}{2}$. Moreover, $e^T C_k e = (m_k - 1)(t-k)^2$, implying

$$e^T x = k\rho_k - (t-k)(m_k - 1), \quad \frac{e^T x}{t-k} + a = \rho_k \left(\frac{k}{t-k} + 2 \right). \tag{36}$$

Taking the inner product of relation (c) with the all-ones vector and using (35)(a) and (36), we find:

$$\begin{aligned} 0 &= e^T Ax + ae^T y = e^T A(\rho_k e - \tfrac{1}{t-k} C_k e) + ae^T y \\ &= \rho_k e^T Ae - \tfrac{1}{t-k} e^T (I - yx^T) e + ae^T y = \rho_k e^T Ae - \tfrac{k}{t-k} + e^T y (\tfrac{x^T e}{t-k} + a) \\ &= \rho_k (e^T Ae + 2e^T y) + \tfrac{k}{t-k} (\rho_k e^T y - 1); \end{aligned}$$

that is,

$$e^T Ae + 2e^T y = \frac{k}{t-k} \left(\frac{1}{\rho_k} - e^T y \right). \tag{37}$$

Using relations (35)(d),(b) and (36), we find:

$$\begin{aligned} 1 &= x^T y + ab = (\rho_k e - \tfrac{1}{t-k} C_k e)^T y + ab \\ &= \rho_k e^T y + \tfrac{b}{t-k} e^T x + ab = \rho_k e^T y + b\rho_k (\tfrac{k}{t-k} + 2); \end{aligned}$$

that is,

$$e^T y = \frac{1}{\rho_k} - b \left(\frac{k}{t-k} + 2 \right). \tag{38}$$

Relations (37) and (38) imply that $e^T (C_{k+1})^{-1} e = e^T Ae + 2e^T y + b = b \frac{t^2}{(t-k)^2}$. By the cofactor rule, $b = \frac{\det C_k}{\det C_{k+1}}$, and the lemma follows. □

Corollary 2. *Let $k \geq 2$ and assume that $C_k(t)$ is nonsingular. Then,*

$$\det C_{k+1}(t) = \frac{2t\rho_k}{t-k} \det C_k(t) - \frac{t^2\rho_k^2}{(t-k+1)^2} \det C_{k-1}(t), \tag{39}$$

after setting $\rho_k := m_k \frac{t-k-1}{2}$.

Proof. Setting $P := \begin{pmatrix} I & -\frac{1}{t-k}e \\ 0 & 1 \end{pmatrix}$, we find that $P^T C_{k+1} P = \begin{pmatrix} C_k & \rho_k e \\ \rho_k e^T & \mu \end{pmatrix}$, after setting $\mu := m_k \frac{t(t-k-1)}{t-k}$. Set $u := (C_k)^{-1}e$ and let $v_1, \ldots, v_{k+1}$ denote the columns of $P^T C_{k+1} P$. Then, $v_{k+1} - \rho_k(\sum_{i=1}^k u_i v_i)$ has all zero entries except the last $(k+1)$-th entry equal to $\mu - \rho_k^2(\sum_{i=1}^k u_i) = m_k \frac{t(t-k-1)}{t-k} - \rho_k^2 e^T (C_k)^{-1} e$. Therefore, we can conclude that

$$\det C_{k+1} = \det P^T C_{k+1} P = \left(\frac{2t\rho_k}{t-k} - \rho_k^2 e^T (C_k)^{-1} e \right) \det C_k. \tag{40}$$

Relation (39) now follows directly from Lemma 5 and (40). □

Lemma 6. *Consider the rational functions $f_1(t) = t-1$, $f_2(t) := \frac{t^2(t-2)(3t-2)}{4(t-1)^2}$ and, for $h = 2, \ldots, k$,*

$$f_{h+1}(t) = \frac{2t\rho_h}{t-h} f_h(t) - \frac{t^2\rho_h^2}{(t-h+1)^2} f_{h-1}(t),$$

and the polynomials $g_1(t) := 1$, $g_2(t) := 3t-2$ and, for $h = 2, \ldots, k$,

$$g_{h+1}(t) = \epsilon_h(t-h) g_h(t) - t(t-h-1) g_{h-1}(t),$$

with $\epsilon_h = 1$ if h is even and $\epsilon_h = 4$ otherwise. As before, $\rho_h := m_h \frac{t-h-1}{2}$.

(i) For $h = 2, \ldots, k+1$, $f_h(t) = \frac{t^{\binom{h+1}{2}-1}(t-h)}{4^{\lfloor h/2 \rfloor}(t-1)^h (t-2)^{h-1} \cdots (t-h+1)^2} g_h(t)$.
(ii) For $1 \leq k \leq 6$, $g_k(t) > 0$ for all $t \geq k$. Moreover, $g_7(8) > 0$.

Proof. The proof for (i) is by induction on k. For (ii), setting $G_k(t) := g_k(t+k)$, one has to show that $G_k(t) > 0$ for $t \geq 0$, $k \leq 6$. This follows from the fact that $G_2(t) = 4 + 3t$, $G_3(t) = 7 + 7t + 2t^2$, $G_4(t) = 64 + 68t + 30t^2 + 5t^3$, $G_5(t) = 167 + 165t + 84t^2 + 25t^3 + 3t^4$, $G_6(t) = 1776 + 1296t + 540t^2 + 248t^3 + 70t^4 + 7t^5$. Moreover, $g_7(8) = 1024$. □

We can now conclude the proof of Theorem 1. Consider $1 \leq r \leq \min(\alpha(G), 6)$ and $t \geq r+1$. We show that (33) holds using Corollary 2 and Lemma 6. First note that $\det C_h(t) = f_h(t)$ for $h = 1, 2$ (direct verification). Let $k \in \{1, \ldots, r\}$. If $k = 1, 2$, then $\det C_k(t) > 0$. Assume $k \geq 3$ and $C_{k-1}(t) \succ 0$. By Corollary 2, $\det C_1(t), \ldots, \det C_k(t)$ are related via (39); that is, $\det C_h(t) = f_h(t)$ for $h = 1, \ldots, k$. We now deduce from Lemma 6 that $\det C_k(t) > 0$. This shows that $C_r(t) \succ 0$ for $t \geq r+1$, which concludes the proof of Theorem 1.

Let us now conclude the proof of Theorem 2 in the case when $\alpha(G) = 8$. We have to show that the matrix $M = t(I + A_G) - J$ from (18) with $t := \alpha(G) = 8$ belongs to $\mathcal{K}_n^{(7)}$. We use the same argument as in the proof of Theorem 1. Thus we are left with the task of proving that $\det C_1(t), \ldots, \det C_7(t) > 0$ for $t = 8$. This follows from the assertions $g_1(8), \ldots, g_6(8), g_7(8) > 0$ in Lemma 6.

Note that the same argument cannot be used for proving Conjecture 1 in the case $\alpha(G) = 9$, since $g_1(9), \ldots, g_6(9) > 0$ while $g_7(9) < 0$ which implies that the matrix $C_7(9)$ is not positive semidefinite.

4 Proof of Theorem 3

Obviously, $las^{(1)} = \vartheta^{(0)}(G)$. In view of (14), we have to show that $las^{(r)} \leq \tilde{\vartheta}^{(r-1)}(G)$ for any positive integer r. For this, let $x \in \mathbb{R}^{\mathcal{P}_{2r}(V)}$ be feasible for (1), i.e., $x_\emptyset = 1$, $x_I \geq 0$ ($|I| = r+1$), $x_{ij} = 0$ ($ij \in E$), and $M_r(x) \succeq 0$. Then, $x_I = 0$ for any $I \in \mathcal{P}_{2r}(V)$ containing an edge. We may assume that $\sum_{i=1}^n x_i > 0$. For $p = 1, \ldots, r+1$, define

$$\ell_p := \sum_{\beta \in I(n,p-1)} \frac{(p-1)!}{\beta!} x_{S(\beta)}.$$

Then, $\ell_1 = 1$, $\ell_p \geq \ell_2 = \sum_{i=1}^n x_i > 0$ for $p \geq 2$. For $p = 1, \ldots, r$, define $y = (y_\delta)_{\delta \in I(n,2p+2)}$ as follows: $y_\delta = 0$ if $S_{odd}(\delta) \neq \emptyset$, $y_\delta := \frac{1}{\ell_p} x_{S(\delta)}$ otherwise (then $|S(\delta)| \leq p+1 \leq r+1$).

Lemma 7. $N_{p+1}(y) \succeq 0$.

Proof. For $I \subseteq V$, set $\mathcal{O}_I := \{\beta \in I(n, p+1) \mid S_{odd}(\beta) = I\}$ and $N_I := (y_{\beta+\beta'})_{\beta,\beta' \in \mathcal{O}_I}$. Then, $N_{p+1}(y)$ is a block diagonal matrix with the matrices N_I ($I \subseteq V$) as diagonal blocks. As $\ell_p N_I = (x_{S(\beta) \cup S(\beta')})_{\beta,\beta' \in \mathcal{O}_I}$, $N_I \succeq 0$ since it is obtained from a principal submatrix of $M_r(x)$ by duplicating certain rows/columns (unless $|I| = r+1$ in which case N_I is the 1×1 matrix with entry $x_{|I|} \geq 0$, implying again $N_I \succeq 0$). □

Therefore, the matrix $Z(p) := C(y) = \sum_{\gamma \in I(n,p-1)} \frac{(p-1)!}{\gamma!} N^\gamma(y)$ belongs to the cone $\mathcal{C}_n^{(p-1)}$. Moreover, $Z(p)_{ij} = 0$ if $ij \in E$. Define the matrix

$$\tilde{Z}(p) := \begin{pmatrix} 1 & Z(p)_{11} \ldots Z(p)_{nn} \\ Z(p)_{11} & \\ \vdots & \quad Z(p) \\ Z(p)_{nn} & \end{pmatrix}. \tag{41}$$

Lemma 8. $\tilde{Z}(p) \succeq 0$.

Proof. The matrix:

$$\ell_p \tilde{Z}(p) = \sum_{\gamma \in I(n,p-1)} \frac{(p-1)!}{\gamma!} \begin{pmatrix} x_{S(\gamma)} & y_{2\gamma+4e_1} \cdots y_{2\gamma+4e_n} \\ y_{2\gamma+4e_1} & \\ \vdots & (y_{2\gamma+2e_j+2e_k})_{j,k=1}^n \\ y_{2\gamma+4e_n} & \end{pmatrix}$$
$$= \sum_{\gamma \in I(n,p-1)} \frac{(p-1)!}{\gamma!} \begin{pmatrix} x_{S(\gamma)} & x_{S(\gamma+e_1)} \cdots x_{S(\gamma+e_n)} \\ x_{S(\gamma+e_1)} & \\ \vdots & (x_{S(\gamma+e_j+e_k)})_{j,k=1}^n \\ x_{S(\gamma+e_n)} & \end{pmatrix}$$

is positive semidefinite, since the matrices in the above summation are principal submatrices of $M_r(x)$. □

Lemma 9. $\sum_{i,j=1}^n Z(p)_{ij} = \frac{\ell_{p+2}}{\ell_p}$ *and* $\sum_{i=1}^n Z(p)_{ii} = \frac{\ell_{p+1}}{\ell_p}$.

Proof. Direct verification. □

Lemma 10. $\frac{\ell_{p+2}}{\ell_{p+1}} \geq \frac{\ell_{p+1}}{\ell_p}$.

Proof. By Lemma 8, $\tilde{Z}(p) \succeq 0$, implying $Z(p) - diag(Z(p))diag(Z(p))^T \succeq 0$. Therefore, $e^T(Z(p) - diag(Z(p))diag(Z(p))^T)e \geq 0$, yielding $\sum_{i,j=1}^n Z(p)_{ij} \geq (\sum_{i=1}^n Z(p)_{ii})^2$. The result now follows using Lemma 9. □

From Lemmas 9 and 10, we deduce that $\sum_{i=1}^n Z(r)_{ii} = \frac{\ell_{r+1}}{\ell_r} \geq \frac{\ell_2}{\ell_1} = \sum_{i=1}^n x_i$. As the matrix $Z(r)$ is feasible for the program (13) defining the parameter $\tilde{\vartheta}^{(r)}(G)$, this shows that $\tilde{\vartheta}^{(r)}(G) \geq \sum_{i=1}^n x_i$ and thus $\tilde{\vartheta}^{(r)}(G) \geq las^{(r)}(G)$, concluding the proof of Theorem 3.

Acknowledgements. The second author thanks Etienne de Klerk for several valuable discussions about the topic of this paper.

References

1. Bomze, I.M., de Klerk, E. 2002. Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. *Journal of Global Optimization* **24** 163–185.
2. Delsarte, P. 1973. *An Algebraic Approach to the Association Schemes of Coding Theory.* [Philips Research Reports Supplements (1973) No. 10] Philips Research Laboratories, Eindhoven.
3. De Klerk, E., Pasechnik, D.V. 2002. Approximating the stability number of a graph via copositive programming. *SIAM Journal on Optimization* **12** 875–892.
4. Lasserre, J.B. 2001. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization* **11** 796–817.

5. Lasserre, J.B. 2001. An explicit exact SDP relaxation for nonlinear $0-1$ programs. In K. Aardal and A.M.H. Gerards, eds., *Lecture Notes in Computer Science* **2081** 293–303.
6. Laurent, M. 2003. A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for $0-1$ programming. *Mathematics of Operations Research* **28** 470–496.
7. Laurent, M. 2005. Strengthened Semidefinite Bounds for Codes. Preprint. Available at `http://www.cwi.nl/~monique`
8. Lovász, L. 1979. On the Shannon capacity of a graph. *IEEE Trans. Inform. Theory* **25** 1–7.
9. Lovász, L., Schrijver, A. 1991. Cones of matrices and set-functions and $0-1$ optimization. *SIAM Journal on Optimization* **1** 166–190.
10. R.J. McEliece, R.J., Rodemich, E.R., Rumsey, H.C., 1978. The Lovász' bound and some generalizations. *Journal of Combinatorics, Information & System Sciences* **3** 1 34–152.
11. Motzkin, T.S., Straus, E.G., 1965. Maxima for graphs and a new proof of a theorem of Túran. *Canadian J. Math.* **17** 533-540.
12. Parrilo, P.A. 2000. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization.* PhD thesis, California Institute of Technology.
13. Pólya, G., 1974. Collected Papers, *MIT Press, Cambridge, Mass., London*, vol. 2, pages 309–313.
14. Reznick, B. Some concrete aspects of Hilbert's 17th problem. Preprint. Available at `http://www.math.uiuc.edu/~reznick/`
15. Reznick, B. 1992. Sums of even powers of real linear forms. *Memoirs of the American Mathematical Society*, Number 463, 1992.
16. Schrijver, A. 1979. A comparison of the Delsarte and Lovász bounds. *IEEE Trans. Inform. Theory* **25** 425–429.
17. Schrijver, A. 2004. New code upper bounds from the Terwiliger algebra. Preprint. Available at `http://www.cwi.nl/~lex`
18. Schweighofer, M. Optimization of polynomials on compact semialgebraic sets. *SIAM Journal on Optimization*, to appear. Available at `http://www.math.uni-konstanz.de/~schweigh/`

Approximation Algorithms for Semidefinite Packing Problems with Applications to MAXCUT and Graph Coloring

G. Iyengar, D.J. Phillips, and C. Stein*

Department of IEOR, Columbia University, New York, NY
{garud, cliff}@ieor.columbia.edu
djp80@columbia.edu

Abstract. We describe the semidefinite analog of the vector packing problem, and show that the semidefinite programming relaxations for MAXCUT [10] and graph coloring [17] are in this class of problems. We extend a method of Bienstock and Iyengar [5] which was based on ideas from Nesterov [25] to design an algorithm for computing -approximate solutions for this class of semidefinite programs. Our algorithm is in the spirit of Klein and Lu [18], and decreases the dependence of the run-time on from $^{-2}$ to $^{-1}$. For sparse graphs, our method is faster than the best specialized interior point methods. A significant feature of our method is that it treats both the MAXCUT and the graph coloring problem in a unified manner.

1 Introduction

Semidefinite programming (SDP) has become a powerful tool for solving optimization problems. Lovász [22] applied semidefinite programming to model the Shannon-capacity of a graph, which, with the work of Grötschel, Lovász, and Schrijver [14], led to the first polynomial-time algorithm for finding the largest stable set in a perfect graph. Beginning with the work of Goemans and Williamson [10], semidefinite programming has been used as a tool for approximating NP-hard optimization problems. In this case, a semidefinite relaxation of the original problem is formulated and solved and then a rounding step is used to output a feasible and approximately optimal solution to the original problem. Since Goemans and Williamson used this technique to design approximation algorithms for MAXCUT, MAXDICUT and MAX2SAT, it has been used successfully by several other researchers. Karger, Motwani, and Sudan [17] use an SDP relaxation and rounding strategy to develop an approximation algorithm for the graph coloring problem. Skutella [30] used SDP to solve a scheduling problem,

* First author partially supported by NSF grants CCR-00-09972, DMS-01-04282 and ONR grant N000140310514. Second author partially supported by NSF Grants DGE-0086390 and DMI-9970063. Third author partially supported by NSF Grant DMI-9970063.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 152–166, 2005.

and recently, Arora, Rao and Vazirani [2] have used semidefinite programming to approximate graph partitioning problems. In all these problems, using the SDP relaxation yields better approximation bounds than using a linear programming relaxation or combinatorial techniques. In this paper we show that several of these and other SDPs appearing in the context of relaxations of combinatorial optimization problems can be viewed as *packing* problems over semidefinite matrices. We will call the SDP

$$\begin{aligned} &\max \mathbf{C} \bullet \mathbf{X}, \\ &\text{s.t. } \mathbf{A}_i \bullet \mathbf{X} \leq 1, \quad i = 1, \ldots, m, \\ &\quad \mathbf{X} \in \mathcal{X} \subseteq \mathbf{R}^{n \times n}, \end{aligned} \tag{1}$$

an SDP with *packing* constraints if the following is true:

(i) Each $\mathbf{A}_i \succeq \mathbf{0}$ ($\mathbf{A} \succeq \mathbf{0}$ denotes that $\mathbf{A}$ is symmetric positive definite);
(ii) Optimizing a linear function over the set $\mathcal{X} \subseteq \{\mathbf{X} : \mathbf{X} \succeq \mathbf{0}\}$ is "easy"; e.g., let $\mathcal{X} = \{\mathbf{X} : \mathbf{X} \succeq \mathbf{0}, \mathbf{Tr}(\mathbf{X}) = a > 0\}$. Then $\max\{\mathbf{C} \bullet \mathbf{X} : \mathbf{X} \in \mathcal{X}\} = a \max\{\lambda_{\max}(\mathbf{C}), 0\}$, where $\lambda_{\max}(\mathbf{C})$ denotes the maximum eigenvalue of $\mathbf{C}$.

The SDP with packing constraints is the natural extension of a vector problem with packing constraints [27] defined as follows.

$$\begin{aligned} &\max \mathbf{c}^T \mathbf{x}, \\ &\text{s.t. } \mathbf{a}_i^T \mathbf{x} \leq 1, \quad i = 1, \ldots, m, \\ &\quad \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{2}$$

where $\mathcal{X}$ is a polytope over which linear optimization is easy. The vector packing (and the corresponding covering) problem has received a lot of attention in the combinatorial optimization community. Since (2) is a linear program (LP), an optimal solution can be found in polynomial time. However, in practice, for common applications such as multicommodity flow, it takes an extremely long time and large amount of memory to solve these problems to optimality [4]. On the other hand, if one is content with a solution within a $(1 + \epsilon)$ factor of the optimal solution (called an ϵ-optimal solution), the situation is much more encouraging. Leveraging the fact that LPs over $\mathcal{X}$ are "easy," one designs Lagrangian relaxation algorithms which "dualize" the constraints $\mathbf{a}_i^T \mathbf{x}$ with appropriate multipliers and reduce the packing problem to a series of LPs over $\mathcal{X}$. These ϵ-optimal approximation algorithms tend to be faster, both in theory and in practice. Although the computational complexity of these algorithms depends on many parameters defining the problem size, here we will focus mainly on the dependence of the number of iterations on ϵ. Shahrokhi and Matula [29] developed an algorithm that computes an ϵ-optimal solution for the concurrent flow problem (a special case of (2)) in $O(\epsilon^{-7})$ iterations, where each iteration involves optimizing a linear function over $\mathcal{X}$. Subsequent research quickly reduced the dependence on ϵ to $O(\epsilon^{-2})$, and many other improvements reduced the number of iterations, the time per iteration, and expanded the techniques to broader packing and covering problems. See, e.g., [20, 21, 12, 13, 27, 28, 9, 8] for details. A recent breakthrough by Bienstock and Iyengar [5] employed a result

of Nesterov [25] to reduce the dependence on ϵ to $O(\epsilon^{-1})$. Since the basic operation in all of these algorithms is linear optimization ([5] considers a regularized version), all of these methods are useful only when this step is cheap.

In this paper, we continue research into algorithms which have $O(\epsilon^{-1})$ dependence on ϵ, and extend the theory to SDPs with packing constrains. The main contributions in this paper are as follows.

(a) We show that several interesting SDPs, such as the MAXCUT SDP, the graph coloring SDP, and the Lovász Shannon-capacity problem can all be cast as SDPs with packing constraints. This allows us to design algorithms for all these problems in a unified manner, leveraging the knowledge gained from designing algorithms for the vector packing problem.
(b) We extend the technique proposed by Nesterov [25] to design an algorithm for the SDP with packing constraints that computes an ϵ-optimal solution in $O(\frac{n \log n}{\epsilon})$ iterations, where each iteration solves a regularized linear optimization problem over the set $\mathcal{X} = \{\mathbf{X} : \mathbf{X} \succeq \mathbf{0}, \mathbf{Tr}(\mathbf{X}) \leq 1\}$, i.e. we reduce the SDP packing problem to a series of simple optimization problems over the set $\mathcal{X}$. As in the case of vector packing, such an algorithm will be attractive if linear optimization over $\mathcal{X}$ is cheap.
(c) Algorithms for vector packing problems, including the one in [5], do *not* yield a feasible solution (one notable exception is in [7]); instead, they compute an ϵ-feasible solution that is close to optimal. In contrast, the algorithm proposed in this paper computes ϵ-optimal *strictly* feasible solution for two special instances, namely the MAXCUT SDP and the graph coloring SDP. This feature is also true of the algorithms presented by Klein and Lu [18] for MAXCUT and coloring (see below).
(d) We show that a regularized version of the linear optimization problem $\max\{\mathbf{W} \bullet \mathbf{X} : \mathbf{X} \in \mathcal{X}\}$, with $\mathbf{W} = \mathbf{W}^T$, can be solved in $O\left(n(n+m)\log^3(1/\epsilon)\right)$ time, where m denotes the maximum number of non-zero terms in $\mathbf{W}$.

Klein and Lu [18] (see also [19]) describe ϵ-approximation algorithms to solve the MAXCUT SDP and the graph coloring SDP that build on an algorithm described in [27]. The number of iterations required to compute ϵ-optimal solution to the MAXCUT SDP (resp. graph coloring SDP) is $O(\epsilon^{-2} nm \log^2 n)$ (resp. $O(\epsilon^{-4} nm \log^3 n)$). When graphs are sparse, algorithms in [18] are especially efficient. This work, however, does not utilize the fact that both the MAXCUT and the graph coloring SDPs are, in fact, particular instances of a more general problem class that can be efficiently solved. (Note the large difference in the run times for MAXCUT and graph coloring.)

Interior point algorithms can solve SDPs with packing constraints in time polynomial in the input size and logarithmic in the error ϵ [1, 26]. Specialized interior point methods [3, 6] for solving the MAXCUT SDP have a worst-case complexity of $O(n^{3.5} \log(\frac{1}{\epsilon}))$; in practice, however, the specialized methods perform faster than this worst-case bound. The theoretical complexity of general interior point methods for solving the graph coloring problem is $O(n^{6.5} \log(\frac{1}{\epsilon}))$.

The significant difference in worst-case complexity is a consequence of the fact that the number of constraints increases from $O(n)$ to $O(n^2)$. No specialized interior point methods have been developed for graph coloring.

In contrast to interior point methods, the iteration count of the algorithm proposed in this paper is $O(\frac{n \log p}{\epsilon})$ where p is the number of constraints. We assume that $p = O(n^k)$ for some fixed k, as is typically the case, which results in an iteration count of $O(\frac{n \log n}{\epsilon})$. For MAXCUT, $k = 1$, and for graph coloring $k = 1.5$. Since any linear optimization problem over $\mathcal{X}$ can be solved by computing a spectral decomposition, each iteration in the packing algorithm is $O(n^3)$, yielding a worst case bound of $O(\frac{n^4 \log n}{\epsilon})$. If the matrix is sparse, i.e., $m = O(n)$, we obtain a bound of $O(\epsilon^{-1} n^3 \log(n) \log^3(\frac{1}{\epsilon}))$. Clearly, interior point methods will be superior to our algorithm for very small ϵ; thus, these methods will be competitive only for moderately small ϵ or in the presence of sparsity. In addition, as is the case with vector packing problem, the algorithms proposed here are interesting only for large problems with special structure that still allow cheap linear optimization over $\mathcal{X}$ but the interior point methods are not able to leverage the structure to reduce memory requirements.

Our presentation focuses on the MAXCUT and coloring problems, and briefly describes the Lovász Shannon-capacity problem. Our results apply to a broader class of SDPs but we do not pursue the details in this extended abstract. Section 2 introduces notations and definitions. In Section 3, we describe the special case of the MAXCUT SDP and how solutions to this SDP can be computed from ϵ-optimal solutions to a related saddle-point problem. In Section 4, we describe the main algorithm that approximates the saddle-point problem. In Section 5, we describe how the graph coloring SDP can be approximated in an analogous fashion. These two instances will clearly imply the algorithm for general SDP packing problems. We are currently simplifying some of the details of the general algorithm. In Section 6, we describe the Lovász Shannon-capacity problem as an SDP with packing constraints. In Section 7, we describe how to exploit sparsity to improve the run-time of a bottleneck subroutine used to optimize over $\mathcal{X}$. The technical details for the algorithms approximating the MAXCUT and graph coloring SDPs can be found in [16].

2 Notation and Definitions

All vectors will be denoted by lowercase boldfaced letters, and matrices by capital boldfaced letters. Unless explicitly indicated, we use n dimensional column vectors and $n \times n$ matrices. We use $\mathbf{I}$ to denote the identity matrix and $\mathbf{1}$ for the vector of all ones. For a square matrix, $\mathbf{A} = [a_{ij}]$, $\mathbf{diag}(\mathbf{A}) = [a_{11}\ a_{22}\ \ldots\ a_{nn}]^T$ denotes the main diagonal of $\mathbf{A}$. For a vector $\mathbf{a}$, $\mathbf{diag}(\mathbf{a}) = [d_{ij}]$, where $d_{ij} = a_i$ when $i = j$ and zero when $i \neq j$, i.e. $\mathbf{diag}(\mathbf{a})$ is the diagonal matrix with the vector $\mathbf{a}$ as the main diagonal. For matrices $\mathbf{A}$ and $\mathbf{B}$, we define $\mathbf{A} \bullet \mathbf{B} = \mathbf{Tr}(\mathbf{AB})$, and use $\mathbf{A} \succeq \mathbf{0}$ to indicate that $\mathbf{A}$ is a positive semidefinite matrix.

For a function $\Phi : \Theta \times \Upsilon \to \mathbf{R}$ consider the *saddle-point* problem

$$\max_{\mathbf{z}\in\Theta} \min_{\mathbf{p}\in\Upsilon} \Phi(\mathbf{z}, \mathbf{p}). \tag{3}$$

For a given $\epsilon > 0$, we say that the pair $(\bar{\mathbf{z}}, \bar{\mathbf{p}}) \in \Theta \times \Upsilon$ is an *ϵ-saddle-point* if,

$$0 \leq \min_{\mathbf{p}\in\Upsilon} \Phi(\bar{\mathbf{z}}, \mathbf{p}) - \max_{\mathbf{z}\in\Theta} \Phi(\mathbf{z}, \bar{\mathbf{p}}) \leq \epsilon. \tag{4}$$

Let $h : \Theta \to \mathbf{R}$ be a function that we wish to minimize and let h^* denote the minimum value. We say that $\bar{\mathbf{z}}$ is ϵ-optimal in the absolute sense if $h(\bar{\mathbf{z}}) \leq h^* + \epsilon$, i.e. $h(\bar{\mathbf{z}})$ is within an *additive* error ϵ to the optimal value. Suppose $h^* \geq C$. Then $h(\bar{\mathbf{z}}) \leq h^* + \epsilon = h^* + C(\epsilon/C) \leq (1 + \epsilon/C)h^*$, thus, an ϵ-optimal solution in the absolute sense has a *relative* error at most ϵ/C. If C is a constant, then an ϵ-optimal solution in the absolute sense is an ϵ-optimal solution in the more traditional multiplicative sense. (We will make analogous definitions for maximization problems.) This relation between the absolute and relative error in each of these problems was described by Klein and Lu [18]. Our algorithms will actually return an ϵ-optimal primal-dual pair, i.e. a pair of primal and dual solutions whose objective values differ by no more than ϵ. By standard strong duality arguments, this immediately implies that both the primal and the dual are ϵ-optimal.

Let $\boldsymbol{\gamma} \in \mathbf{R}^n$. Then the following linear program has a simple solution.

$$\max_{\mathbf{y}} \Big\{ \sum_{i=1}^{n} \gamma_i y_i : \mathbf{y} \geq 0, \sum_{i=1}^{n} y_i \leq 1 \Big\} = \max\big\{0, \max_{1\leq i\leq n} \{\gamma_i\}\big\}. \tag{5}$$

For a square matrix $\mathbf{A}$, $\lambda_1(\mathbf{A}) \leq \lambda_2(\mathbf{A}) \leq \ldots \leq \lambda_n(\mathbf{A})$ will denote the ordered set eigenvalues of $\mathbf{A}$, $\lambda_{\max}(\mathbf{A}) := \lambda_n(\mathbf{A})$ and $\lambda_{\min} := \lambda_1(\mathbf{A})$. Note that for all square matrices $\mathbf{A}$, the following optimization problem reduces to a linear program identical to (5) in the space of eigenvalues:

$$\begin{aligned} \max_{\mathbf{X}} \big\{ \mathbf{A} \bullet \mathbf{X} : \mathbf{Tr}(\mathbf{X}) \leq 1, \mathbf{X} \succeq 0 \big\} &= \max_{\mathbf{y}} \Big\{ \sum_{i=1}^{n} \lambda_i(\mathbf{A}) y_i : \mathbf{y} \geq 0, \sum_i y_i \leq 1 \Big\} \\ &= \max\big\{0, \lambda_{\max}(\mathbf{A})\big\}. \end{aligned} \tag{6}$$

3 The SDP Relaxation for MAXCUT

In this section we review the well-known SDP relaxation of the exact vector formulation for the MAXCUT problem. We then show the equivalence of this relaxation to a maximin problem. The exact vector formulation for MAXCUT is as follows.

$$\begin{aligned} \max \ & \tfrac{1}{4}\mathbf{W} \bullet (\mathbf{1}\mathbf{1}^T - \mathbf{X}) \\ \text{subject to } & \mathbf{X} = \mathbf{x}\mathbf{x}^T, \\ & \mathbf{x} \in \{-1, 1\}^n, \end{aligned} \tag{7}$$

where $\mathbf{W}$ is the weight matrix. The *Laplacian* $\mathbf{L} = [\ell_{ij}]$ of a weighted graph with weights $\mathbf{W}$ is given by

$$\ell_{ij} = \begin{cases} -w_{ij}, & i \neq j, \\ \sum_{k=1}^{n} w_{ik}, & i = j. \end{cases} \tag{8}$$

The Laplacian $\mathbf{L}$ of a graph is a positive semidefinite matrix. From (8) it follows that (7) is equivalent to

$$\begin{aligned} \max \ & \tfrac{1}{4}\mathbf{L} \bullet \mathbf{X} \\ \text{subject to } & \mathbf{X} = \mathbf{x}\mathbf{x}^T \\ & \mathbf{x} \in \{-1, 1\}^n, \end{aligned} \tag{9}$$

The SDP relaxation of (9) employed by the Goemans-Williamson [10] approximation algorithm is equivalent to the following

$$\begin{aligned} \max \ & \mathbf{L} \bullet \mathbf{X} \\ \text{subject to } & \mathbf{diag}(\mathbf{X}) \leq \mathbf{1}, \\ & \mathbf{X} \in \bar{\mathcal{X}} \equiv \{\mathbf{X} : \mathbf{X} \succeq \mathbf{0}, \mathbf{Tr}(\mathbf{X}) \leq n\}. \end{aligned} \tag{10}$$

Note that relaxing $\mathbf{diag}(\mathbf{X}) = \mathbf{1}$ to $\mathbf{diag}(\mathbf{X}) \leq \mathbf{1}$ does not change the formulation, because increasing the diagonal of a positive semidefinite matrix keeps it positive semidefinite, and only increases the objective. The extra constraint $\mathbf{Tr}(\mathbf{X}) \leq n$ in the definition of $\mathcal{X}$ is implied by $\mathbf{diag}(\mathbf{X}) \leq \mathbf{1}$. Since linear optimization over $\bar{\mathcal{X}}$ reduces to a problem of the form (6), it follows that (10) is an SDP with packing constraints. We assume that the edge weights $\{w_{ij}\}$ sum to 1, i.e. $\mathbf{L} \bullet \mathbf{I} = 2$. Since $\mathbf{I}$ is feasible for (10), it follows that the optimal value of (10) is at least 2.

On dualizing the constraints $\mathbf{diag}(\mathbf{X}) - \mathbf{1}$ we get the saddle-point problem

$$\max_{\{\mathbf{X}:\mathbf{X}\succeq\mathbf{0},\mathbf{Tr}(\mathbf{X})\leq n\}} \min_{\{\mathbf{u}:\mathbf{u}\geq\mathbf{0}\}} \Big\{\mathbf{L} \bullet \mathbf{X} - \sum_{i=1}^{n} u_i(x_{ii} - 1)\Big\}. \tag{11}$$

We want to compute a good solution $\widehat{\mathbf{X}}$ for (10) by starting from an initial $\mathbf{X}^{(0)} \succeq \mathbf{0}$ with $\mathbf{Tr}(\mathbf{X}^{(0)}) \leq n$, and then iterating by choosing the next dual iterate $\mathbf{u}^{(k+1)}$ (resp. primal iterate $\mathbf{X}^{(k+1)}$) to be the "best response" to current primal iterate $\mathbf{X}^{(k)}$ (resp. dual iterate $\mathbf{u}^{(k)}$). However, this is impossible unless we are able to bound the "width" of the set of dual variables. In Theorem 1, we show that it is sufficient to restrict to dual variables $\mathbf{u}$ such that $\sum_{i=1}^{n} u_i \leq 5n$. We will find it more convenient to work with the following scaled version of the saddle-point problem (11):

$$\max_{\mathbf{X}\in\mathcal{X}} \min_{\mathbf{u}\in\mathcal{U}} \phi(\mathbf{X}, \mathbf{u}) = \min_{\mathbf{u}\in\mathcal{U}} \max_{\mathbf{X}\in\mathcal{X}} \phi(\mathbf{X}, \mathbf{u}),$$

where

$$\phi(\mathbf{X}, \mathbf{u}) = n\mathbf{X} \bullet \mathbf{L} - \sum_{i=1}^{n} 5nu_i(nx_{ii} - 1), \tag{12}$$

$$\mathcal{X} = \Big\{\mathbf{X} \in \mathbf{R}^{n\times n} : \mathbf{X} \succeq 0, \mathbf{Tr}(\mathbf{X}) \leq 1\Big\}, \tag{13}$$

$$\mathcal{U} = \Big\{\mathbf{u} \geq 0 : \sum_{i=1}^{n} u_i \leq 1\Big\}. \tag{14}$$

Theorem 1. *Fix $\epsilon > 0$. Suppose $(\bar{\mathbf{X}}, \bar{\mathbf{u}}) \in \mathcal{X}\times\mathcal{U}$ is an ϵ-saddle-point with respect to ϕ satisfying (4). Let $\bar{d} = n\max_{1\leq i\leq n}\{\bar{x}_{ii}\}$, $\bar{\lambda} = \lambda_{\max}(\mathbf{L} - 5n\,\mathbf{diag}(\bar{\mathbf{u}}))$, and*

$$\widehat{\mathbf{X}} = \begin{cases} n\bar{\mathbf{X}} & \bar{d} \leq 1, \\ n\bar{\mathbf{X}}/\bar{d}, & \textit{otherwise;} \end{cases} \qquad \widehat{\mathbf{u}} = \begin{cases} 5n\bar{\mathbf{u}}, & \bar{\lambda} \leq 0, \\ 5n\bar{\mathbf{u}} + \bar{\lambda}\mathbf{1}, & \textit{otherwise.} \end{cases} \tag{15}$$

Then $(\widehat{\mathbf{X}}, \widehat{\mathbf{u}})$ are an ϵ-optimal primal-dual pair for (10) and (16).

Proof. From the definition of $\bar{d}$, it follows that $\widehat{\mathbf{X}}$ is feasible for (10). The dual of (10) is given by

$$\begin{array}{c} \min \sum_{i=1}^{n} u_i \\ \text{subject to } \mathbf{diag}(\mathbf{u}) - \mathbf{L} \succeq \mathbf{0}, \\ \mathbf{u} \geq 0. \end{array} \tag{16}$$

Since

$$\mathbf{diag}(\widehat{\mathbf{u}}) - \mathbf{L} = 5n\,\mathbf{diag}(\bar{\mathbf{u}}) + \max\big\{0, \lambda_{\max}(\mathbf{L} - \mathbf{diag}(5n\bar{\mathbf{u}}))\big\}\mathbf{I} - \mathbf{L} \succeq \mathbf{0},$$

it follows that $\widehat{\mathbf{u}}$ is dual feasible. From the definition of $\mathcal{X}$ and (6) it follows that

$$\begin{aligned} \max_{\mathbf{X}\in\mathcal{X}} \phi(\mathbf{X}, \bar{\mathbf{u}}) &= \max_{\mathbf{X}\in\mathcal{X}} \Big\{n\mathbf{X} \bullet (\mathbf{L} - 5n\,\mathbf{diag}(\bar{\mathbf{u}})) + 5n\sum_{i=1}^{n}\bar{u}_i\Big\} \\ &= 5n\sum_{i=1}^{n}\bar{u}_i + n\max\big\{0, \lambda_{\max}\big(\mathbf{L} - 5n\,\mathbf{diag}(\bar{\mathbf{u}})\big)\big\} = \sum_{i=1}^{n}\widehat{u}_i. \end{aligned} \tag{17}$$

From the definition of $\mathcal{U}$ and (5), we have that $\min_{\mathbf{u}\in\mathcal{U}} \phi(\bar{\mathbf{X}}, \mathbf{u}) = n\bar{\mathbf{X}} \bullet \mathbf{L} - \max_{\mathbf{u}\in\mathcal{U}} \Big\{5n\sum_{i=1}^{n} u_i(n\bar{x}_{ii} - 1)\Big\} = n\bar{\mathbf{X}} \bullet \mathbf{L} - 5n\max\big\{0, \bar{d} - 1\big\}$. We show below that $\mathbf{L} \bullet \widehat{\mathbf{X}}$ is at least $\min_{\mathbf{u}\in\mathcal{U}} \phi(\bar{\mathbf{X}}, \mathbf{u})$.

(i) $\bar{d} \leq 1$: Then $\mathbf{L}\bullet\widehat{\mathbf{X}} = n\mathbf{L}\bullet\bar{\mathbf{X}} = n\mathbf{L}\bullet\bar{\mathbf{X}} - 5n\max\big\{0, \bar{d}-1\big\} = \min_{\mathbf{u}\in\mathcal{U}} \phi(\bar{\mathbf{X}}, \mathbf{u})$.

(ii) $\bar{d} > 1$: Since $\mathbf{Tr}(\bar{\mathbf{X}}) \leq 1$, we have $|\widehat{x}_{ij}| \leq \sqrt{\bar{x}_{ii}\bar{x}_{jj}} \leq 1$, and $0 \leq \mathbf{L} \bullet \bar{\mathbf{X}} \leq \sum_{i,j=1}^{n} |\ell_{ij}||\bar{x}_{ij}| \leq \sum_{i,j=1}^{n} |\ell_{ij}| \leq 5$. For all $d > 0$, we have $1/d \geq 1 - (d-1)$, therefore,

$$\begin{aligned} \mathbf{L} \bullet \widehat{\mathbf{X}} = \frac{n\mathbf{L} \bullet \bar{\mathbf{X}}}{\bar{d}} &\geq n\mathbf{L} \bullet \bar{\mathbf{X}} - n(\bar{d} - 1)(\mathbf{L} \bullet \bar{\mathbf{X}}) \\ &\geq n\mathbf{L} \bullet \bar{\mathbf{X}} - 5n(\bar{d} - 1) = \min_{\mathbf{u}\in\mathcal{U}} \phi(\bar{\mathbf{X}}, \mathbf{u}). \end{aligned} \tag{18}$$

From (4), (17), and (18) we have that $\sum_{i=1}^{n} \widehat{u}_i - \mathbf{L} \bullet \widehat{\mathbf{X}} \leq \epsilon$, i.e. $(\widehat{\mathbf{X}}, \widehat{\mathbf{u}})$ is an ϵ-optimal primal-dual pair.

4 Computing the Approximate Saddle-Point $(\bar{\mathbf{X}}, \bar{\mathbf{u}})$

In order to "implement" Theorem 1 we need to compute an ϵ-saddle-point, $(\bar{\mathbf{X}}, \bar{\mathbf{u}})$, for ϕ. In this section we show how to compute an ϵ-saddle point by suitably extending a technique proposed in Nesterov [25]. Due to space constraints, some of the technical details are relegated to [16].

Let $\lambda_i = \lambda_i(\mathbf{L} - 5n\,\mathbf{diag}(\mathbf{u})), i = 1, \ldots, n$, and define $f : \mathcal{U} \to \mathbf{R}$ as follows.

$$f(\mathbf{u}) = \max_{\mathbf{X} \in \mathcal{X}} \phi(\mathbf{X}, \mathbf{u}) = 5n \sum_{i=1}^{n} u_i + n \max\left\{0, \lambda_{\max}\right\}, \tag{19}$$

where $\lambda_{\max} = \lambda_n = \max_{1 \leq i \leq n}\{\lambda_i\}$, and the second equality follows from (6). We compute a pair $(\bar{\mathbf{X}}, \bar{\mathbf{u}})$ such that $f(\bar{\mathbf{u}}) - \min_{\mathbf{u} \in \mathcal{U}} \phi(\bar{\mathbf{X}}, \mathbf{u}) \leq \epsilon$, i.e. it is an ϵ-saddle-point.

Our algorithm is based on a technique developed by Nesterov [25]. Since evaluating f involves solving an LP (see (6)), it is not differentiable. Therefore, we replace it by a smooth approximation f_α (see (20) below). This approach is very similar to the approach taken in the packing-covering literature (see, e.g. [27, 12, 5, 7]). In each iteration t, the algorithm computes a primal iterate $\mathbf{X}^{(t)} \in \mathcal{X}$ such that the gradient $\nabla f_\alpha(\mathbf{u}^{(t)}) = 5n\big(\mathbf{1} - n\,\mathbf{diag}(\mathbf{X}^{(t)})\big)$. This gradient is then used to compute the next dual iterate. The ϵ-saddle-point $(\bar{\mathbf{X}}, \bar{\mathbf{u}})$ is a weighted combination of all the primal-dual iterates generated by the algorithm.

Define f_α as follows.

$$f_\alpha(\mathbf{u}) = 5n \sum_{i=1}^{n} u_i + \frac{1}{\alpha} \ln\left(1 + \sum_{i=1}^{n} e^{\alpha n \lambda_i}\right). \tag{20}$$

Straightforward calculations show that for any $\alpha > 0$, $f(\mathbf{u}) \leq f_\alpha(\mathbf{u}) \leq f(\mathbf{u}) + \frac{\ln(n+1)}{\alpha}$. Therefore, by setting $\alpha = \frac{2\log(n+1)}{\epsilon}$, our problem reduces to computing an $\frac{\epsilon}{2}$-optimal solution for f_α.

Let $\mathbf{V}\,\mathbf{diag}(\boldsymbol{\lambda})\mathbf{V}^T = \mathbf{L} - 5n\,\mathbf{diag}(\mathbf{u})$ denote the eigendecomposition of $\mathbf{L} - 5n\,\mathbf{diag}(\mathbf{u})$. Then the gradient $\nabla f_\alpha(\mathbf{u})$ of $f_\alpha(\mathbf{u})$ is given by

$$\nabla f_\alpha(\mathbf{u}) = 5n(\mathbf{1} - n\,\mathbf{diag}(\mathbf{X}_u)), \tag{21}$$

where

$$\mathbf{X}_u = \frac{\mathbf{diag}\left(\mathbf{V}\,\mathbf{diag}(e^{n\alpha\boldsymbol{\lambda}})\mathbf{V}^T\right)}{\left(1 + \sum_{i=1}^{n} e^{n\alpha\lambda_i}\right)}, \tag{22}$$

and $\mathbf{diag}(e^{n\alpha\boldsymbol{\lambda}})$ denotes a diagonal matrix with $e^{n\alpha\lambda_i}$ as the i-th entry. Computing this gradient will be the subject of Sect. 7. For now we assume the existence of

a procedure SmoothGrad which takes as input a vector $\mathbf{u}$ and returns $\nabla f_\alpha(\mathbf{u})$ and $\mathbf{X}_u$ as defined in equations (21) and (22).

Once the gradient is explicitly known, a Frank-Wolfe-type gradient descent method takes $O(\epsilon^{-2})$ iterations to compute ϵ-optimal $\bar{\mathbf{u}}$. The method proposed by Klein and Lu [18] can be interpreted as such a first-order method. In order to develop an algorithm in which the number of iterations grows as $O(\epsilon^{-1})$ one has to use a second-order Taylor series expansion and a more involved procedure for the inner loop. For all $\mathbf{u}, \mathbf{u}' \in \mathcal{U}$,

$$f_\alpha(\mathbf{u}) \leq f_\alpha(\mathbf{u}') + \nabla f_\alpha(\mathbf{u}')^T(\mathbf{u} - \mathbf{u}') + \frac{n^2\alpha}{2} \|\mathbf{u} - \mathbf{u}'\|_1^2, \tag{23}$$

where $\|\mathbf{u}\|_1 = \sum_{i=1}^n |u_i|$ denotes the $\mathcal{L}_1$-norm. This result essentially follows from the Cauchy-Schwartz inequality. See [25] or [5] for a proof. We want to compute the iterates by minimizing the second-order bound in (23) and, to minimize the computational complexity of each step, we would like to write the iterate in closed form. Since this is impossible to do when the distance between iterates is measured in terms of the $\mathcal{L}_1$-norm, we replace it by the relative entropy or the Kullback-Leibler (K-L) distance $d(\mathbf{u}, \mathbf{u}')$ defined as follows

$$d(\mathbf{u}, \mathbf{u}') = \sum_{i=1}^{n+1} u_i \log\left(\frac{u_i}{u_i'}\right) \tag{24}$$

where $u_{n+1} = 1 - \sum_{i=1}^n u_i$ and $u'_{n+1} = 1 - \sum_{i=1}^n u_i'$. Equation (42) in [16] shows that $d(\mathbf{u}, \mathbf{u}') \geq \frac{1}{2}\|\mathbf{u} - \mathbf{u}'\|_1^2$. Thus, we get the following bound

$$f_\alpha(\mathbf{u}) \leq f_\alpha(\mathbf{u}') + \nabla f_\alpha(\mathbf{u}')^T(\mathbf{u} - \mathbf{u}') + n^2\alpha d(\mathbf{u}, \mathbf{u}'). \tag{25}$$

By using the Lagrange multipliers (see proof after (37) in [16]), $\operatorname{argmin}_{\mathbf{u}\in\mathcal{U}} \left\{d(\mathbf{u}, \mathbf{u}') + \mathbf{g}^T\mathbf{u}\right\}$ is given by

$$u_i = \frac{u_i' e^{-g_i}}{u'_{n+1} + \sum_{k=1}^n u_k' e^{-g_k}}, \quad i = 1, \ldots, n. \tag{26}$$

where $u'_{n+1} = 1 - \sum_{i=1}^n u_i'$. We define a procedure SmoothOpt that given vectors $\mathbf{u}'$ and $\mathbf{g}$, returns the vector $\mathbf{u}$ defined by (26).

We now have all the ingredients necessary to describe the procedure SmoothApprox displayed in Fig. 1. The algorithm wants to compute iterates that converge to the minimum of f_α over $\mathcal{U}$ by sequentially minimizing the approximate second-order Taylors' series expansion (25). Note that the bound (25) does not use the Hessian of f_α. This is because including the Hessian in the Taylors' series leads to a complicated optimization problem that cannot be solved in closed form. On the other hand, without the Hessian term, (25) is not likely to be a good estimate of the function f_α, at least when the iterates are far away from the minimum.

In SmoothApprox we compensate for the lack of Hessian information by using a technique proposed by Nesterov [25]. This technique computes the estimates $\mathbf{y}^{(k)}$ of the minimizer of f_α by keeping track of two sets of iterates: one set

SmoothApprox($\mathbf{L}$,)

$T \leftarrow 4 \quad \log(\quad +1)/\ ; \qquad \leftarrow 2 \quad \log(\quad +1)/\ ; \quad \mathbf{u}^{(0)} \leftarrow \frac{1}{n+1}\mathbf{1}$

$(\mathbf{g}^{(0)}, \mathbf{X}^{(0)}) \leftarrow$ SmoothGrad$(\mathbf{u}^{(0)})$

$\mathbf{s} \leftarrow \frac{1}{2}\mathbf{g}^{(0)}; \quad \widehat{\mathbf{X}} \leftarrow \mathbf{X}^{(0)}$

$\mathbf{y}^{(0)} \leftarrow$ SmoothOpt$\left(\mathbf{u}^{(0)}, \frac{1}{2n^2\alpha}\mathbf{g}^{(0)}\right)$

for $k \leftarrow 0$ **to** T

do

1 $\quad {}_k \leftarrow 2/(k+3)$

2 $\quad \mathbf{z}^{(k)} \leftarrow$ SmoothOpt$\left(\mathbf{u}^{(0)}, \frac{1}{n^2\alpha}\mathbf{s}\right)$

3 $\quad \mathbf{u}^{(k+1)} \leftarrow {}_k\mathbf{z}^{(k)} + (1 - {}_k)\mathbf{y}^{(k)}.$

4 $\quad (\mathbf{g}^{(k+1)}, \mathbf{X}^{(k+1)}) \leftarrow$ SmoothGrad$(\mathbf{u}^{(k+1)})$

5 $\quad \mathbf{s} \leftarrow \mathbf{s} + \frac{k+2}{2}\mathbf{g}^{(k+1)}$

6 $\quad \widehat{\mathbf{X}} \leftarrow \widehat{\mathbf{X}} + (k+2)\mathbf{X}^{(k+1)}$

7 $\quad \widehat{\mathbf{u}}^{(k+1)} \leftarrow$ SmoothOpt$\left(\mathbf{z}^{(k)}, \frac{k+2}{2n^2\alpha}\mathbf{g}^{(k+1)}\right)$

8 $\quad \mathbf{y}^{(k+1)} \leftarrow {}_k\widehat{\mathbf{u}}^{(k+1)} + (1 - {}_k)\mathbf{y}^{(k)}$

$\bar{\mathbf{u}} \leftarrow \mathbf{y}^{(T)} \quad \bar{\mathbf{X}} \leftarrow \frac{2}{(T+1)(T+2)}\widehat{\mathbf{X}}$

return $(\bar{\mathbf{X}}, \bar{\mathbf{u}})$

Fig. 1. Procedure SmoothApprox($\mathbf{L}$,)

uses the gradient $\nabla f_\alpha(\mathbf{u}^{(k)})$ computed at the current iterate, and the other set uses a weighted combination of all the previous iterates $\nabla f_\alpha(\mathbf{u}^{(i)})$, $i = 0, \ldots, k$. From Line 5 of SmoothApprox, we have that at the beginning of iteration k of the variable $\mathbf{s} = \mathbf{s} + \left(\frac{k+1}{2}\right)\mathbf{g}^{(k+1)} = \sum_{i=1}^{k}\left(\frac{i+1}{2}\right)$. Therefore, the iterates $\mathbf{z}^{(k)}$ computed in Line 2 depend on a weighted combination of the gradients at all the previous iterates. The iterate $\widehat{\mathbf{u}}^{(k+1)}$ computed in Line 7 "corrects" the iterate $\mathbf{z}^{(k)}$ using the "local" gradient information $\mathbf{g}^{(k+1)}$ from the current $\mathbf{u}^{(k+1)}$. The new estimate $\mathbf{y}^{(k+1)}$ computed in Line 8 is a convex combination of the previous iterate $\mathbf{y}^{(k)}$ and the iterate $\widehat{\mathbf{u}}^{(k+1)}$.

By Lemma 1 in [16] (proved via results from [25]), the following theorem is immediate:

Theorem 2. *For any $\epsilon > 0$, the output $(\bar{\mathbf{X}}, \bar{\mathbf{u}})$ of* SmoothApprox *is an ϵ-saddle-point. The running time is $O(\epsilon^{-1}Q(n)n\log n)$, where $Q(n)$ is the running time of* SmoothGrad.

Corollary 1. *Suppose* SmoothGrad *computes $\mathbf{X}_u$ in (22) via an eigendecomposition. Then the running time of* SmoothGrad *is $O(\epsilon^{-1}n^4\log n)$.*

5 Semidefinite Relaxation of Graph Coloring

The SDP relaxation for graph coloring corresponding to a graph $\mathcal{G} = (V, E)$ can be formulated as

$$\begin{aligned} &\max \zeta, \\ &\text{s.t. } x_{ii} \leq 1, \quad i = 1, \ldots, n, \\ &\quad x_{ij} + \zeta \leq 0, \quad (i,j) \in E, \\ &\quad \mathbf{X} \in \bar{\mathcal{X}} \equiv \{\mathbf{X} : \mathbf{X} \succeq \mathbf{0}, \mathbf{Tr}(\mathbf{X}) \leq n\}. \end{aligned} \tag{27}$$

For k-colorable graphs ($k \geq 2$), $1 \geq \zeta^* \geq \frac{1}{k}$, hence, an ϵ-optimal solution has a relative error of $k\epsilon$ [17, 18]. Since, without loss of generality, $x^*_{ii} = 1$ in any optimal solution of (27), the second constraint in (27) can be reformulated as $\frac{1}{2}(x_{ii} + x_{jj}) + x_{ij} + \zeta \leq 1$. With this reformulation, the problem (27) reduces to an SDP with packing constraints. The dual of this SDP is given by

$$\begin{aligned} &\min \textstyle\sum_{i=1}^{n} u_i \\ &\text{s.t. } \textstyle\sum_{j=1}^{m} v_j = 1, \\ &\quad \mathbf{diag}(\mathbf{u}) + \mathbf{A}\mathbf{v} \succeq \mathbf{0}, \\ &\quad \mathbf{u}, \mathbf{v} \geq \mathbf{0}, \end{aligned} \tag{28}$$

where $m = |E|$ denotes the number of edges in the graph, $\mathbf{A} : \mathbf{R}^m \mapsto \mathbf{R}^{n \times n}$ is the linear operator $\mathbf{A}\mathbf{v} = \frac{1}{2}\sum_{(i,j)\in E} v_{(i,j)}\mathbf{E}_{ij}$, and $\mathbf{E}_{ij} \in \mathbf{R}^{n\times n}$ with 1's in the (i,j) and (j,i) position and zeros everywhere else.

As in the case with MAXCUT, we dualize both the constraints $\mathbf{diag}(\mathbf{X}) \leq \mathbf{1}$ and $\max_{(i,j)\in E}\{x_{ij}\} + \zeta \leq 0$ to define a saddle-point problem:

$$\max_{\mathbf{X}\in\mathcal{X}} \min_{(\mathbf{u},\mathbf{v})\in\mathcal{U}\times\mathcal{V}} \phi(\mathbf{X}, \mathbf{u}, \mathbf{v}),$$

where

$$\phi(\mathbf{X}, \mathbf{u}, \mathbf{v}) = n\sum_{i=1}^{m} u_i - n(n\mathbf{u} + \mathbf{A}\mathbf{v}) \bullet \mathbf{X}, \tag{29}$$

$$\begin{aligned} \mathcal{X} &= \{\mathbf{X} : \mathbf{X} \succeq \mathbf{0}, \mathbf{Tr}(\mathbf{X}) \leq 1\}, \\ \mathcal{U} &= \{\mathbf{u} : \mathbf{u} \geq 0, \textstyle\sum_{i=1}^{n} u_i \leq 1\}, \\ \mathcal{V} &= \left\{\mathbf{v} : \mathbf{v} \geq \mathbf{0}, \textstyle\sum_{j=1}^{m} v_j = 1\right\}. \end{aligned} \tag{30}$$

Theorem 3. *Fix $\epsilon > 0$. Suppose $(\bar{\mathbf{X}}, (\bar{\mathbf{u}}, \bar{\mathbf{v}})) \in \mathcal{X} \times (\mathcal{U} \times \mathcal{V})$ is an ϵ-saddle-point. Let $\bar{\zeta} = -n \max_{(i,j)\in E}\{\bar{x}_{ij}\}$, $\bar{d} = n \max_{i=1,\ldots,n}\{\bar{x}_{ii}\}$,*

$$\widehat{\mathbf{u}} = n\bar{\mathbf{u}} - \left(\min\{\lambda_{\min}(\mathbf{diag}(\bar{\mathbf{u}}) + \mathbf{A}\bar{\mathbf{v}}), 0\}\right)\mathbf{1}, \qquad \widehat{\mathbf{v}} = \bar{\mathbf{v}}, \tag{31}$$

and

$$\begin{aligned} \widehat{x}_{ii} &= \begin{cases} n\bar{x}_{ii}, & \bar{d} \leq 1, \\ \frac{n\bar{x}_{ii}}{\bar{d}}, & \bar{d} > 1, \end{cases} \\ \widehat{x}_{ij} &= \begin{cases} n\bar{x}_{ij}, & \bar{d} \leq 1, \bar{\zeta} > 0, \\ \frac{n\bar{x}_{ij}}{\bar{d}}, & \bar{d} > 1, \bar{\zeta} > 0, \\ 0, & \bar{\zeta} \leq 0. \end{cases} \\ \widehat{\zeta} &= -\max_{(i,j)\in E}\{\widehat{x}_{ij}\}. \end{aligned} \tag{32}$$

Then $(\widehat{\zeta}, \widehat{\mathbf{X}})$ and $(\widehat{\mathbf{u}}, \widehat{\mathbf{v}})$ are an ϵ-optimal primal-dual pair for (27) and (28).

Proof. One can check that $(\widehat{\mathbf{u}}, \widehat{\mathbf{v}})$ defined in (31) is feasible for the dual SDP (28). From the definition of $\mathcal{X}$, it follows that

$$\max_{\mathbf{X}\in\mathcal{X}} \phi(X, \bar{\mathbf{u}}, \bar{\mathbf{v}}) = n\sum_{i=1}^{m} \bar{u}_i - n\big(\min\{\lambda_{\min}(\mathbf{diag}(\bar{\mathbf{u}}) + n\mathbf{A}\bar{\mathbf{v}}), 0\}\big) = \sum_{i=1}^{n} \widehat{u}_i. \quad (33)$$

The pair $(\widehat{\zeta}, \widehat{\mathbf{X}})$ defined in (32) is feasible for the primal SDP (27). Next, we show that the objective value $\widehat{\zeta}$ is lower bounded by

$$\min_{(\mathbf{u},\mathbf{v})\in\mathcal{U}\times\mathcal{V}} \phi(\bar{\mathbf{X}}, \mathbf{u}, \mathbf{v}) = \bar{\zeta} - n\max\{\bar{d} - 1, 0\}. \quad (34)$$

(a) $\bar{d} \le 1, \bar{\zeta} > 0$: In this case $(\widehat{\zeta}, \widehat{\mathbf{X}}) = (\bar{\zeta}, \bar{\mathbf{X}})$, and $\widehat{\zeta} = \bar{\zeta} = \bar{\zeta} - n\max\{\bar{d} - 1, 0\}$, since $\bar{d} \le 1 \Rightarrow \max\{\bar{d} - 1, 0\} = 0$.

(b) $\bar{d} \ge 1, \bar{\zeta} > 0$: In this case, $\widehat{\zeta} = \frac{\bar{\zeta}}{\bar{d}} \ge \bar{\zeta}\big(1 - (\bar{d} - 1)\big) \ge \bar{\zeta} - n(\bar{d} - 1)$, where the first inequality follows from the fact that $\frac{1}{d} \ge 1 - (d - 1)$ for all $d > 0$, and the second inequality follows from the bound $|\bar{x}_{ij}| \le \sqrt{\bar{x}_{ii}\bar{x}_{jj}} \le 1$.

(c) $\bar{\zeta} \le 0$. In this case, $\widehat{\zeta} = 0 \ge \bar{\zeta} + n\max\{\bar{d} - 1, 0\}$.

Thus, we have established the existence of an ϵ-optimal primal-dual pair $(\widehat{\lambda}, \widehat{\mathbf{X}})$ and $(\widehat{u}, \widehat{v})$.

As in Sect. 3 we work with the function $f(\mathbf{u}, \mathbf{v}) = \max_{\mathbf{X}\in\mathcal{X}} \phi(\mathbf{X}, \mathbf{u}, \mathbf{v})$, and smooth it to obtain

$$f_\alpha(\mathbf{u}, \mathbf{v}) = n\sum_{i=1}^{n} u_i - \frac{1}{\alpha}\log\Big(1 + \sum_{i=1}^{n} e^{-n\alpha\lambda_i}\Big),$$

where $\lambda_i = \lambda_i(n\,\mathbf{diag}(\mathbf{u}) + \mathbf{A}\mathbf{v}), i = 1, \ldots, n$. As before,

$$f(\mathbf{u}, \mathbf{v}) \le f_\alpha(\mathbf{u}, \mathbf{v}) \le f(\mathbf{u}, \mathbf{v}) + \frac{\log(n+1)}{\alpha},$$

and

$$\begin{bmatrix} \nabla_u f_\alpha(\mathbf{u}, \mathbf{v}) \\ \nabla_v f_\alpha(\mathbf{u}, \mathbf{v}) \end{bmatrix} = \begin{bmatrix} n^2\,\mathbf{diag}(\mathbf{X}_u) - n\mathbf{1} \\ \mathbf{A}^T\mathbf{X}_u \end{bmatrix}, \qquad \mathbf{X}_u = \frac{e^{-n\alpha(n\,\mathbf{diag}(\mathbf{u}) + \mathbf{A}\mathbf{v})}}{\big(1 + \sum_{i=1}^{n} e^{-n\alpha\lambda_i}\big)}. \quad (35)$$

The function f_α can be optimized using a procedure very similar to SmoothApprox except that each iteration we solve two optimization problems: one in $\mathbf{u}$ and the other in $\mathbf{v}$. This change is reflected in the timebound by an extra $\log^{.5} m$ factor. Further details of the procedure SmoothApproxColoring are in Appendix B of [16].

Theorem 4. *For any $\epsilon > 0$, the output $(\bar{\mathbf{X}}, \bar{\mathbf{u}}, \bar{\mathbf{v}})$ of* SmoothApproxColoring *is an ϵ-saddle-point. The running time is $O(\epsilon^{-1}Q(n)n\sqrt{\log(n)\log(m)})$, where $Q(n)$ is the running time of* SmoothGrad.

The $\sqrt{\log(m)}$ factor appears because the number of constraints in the graph coloring problem is m as opposed to n in the case of the Maxcut problem. Again the bottleneck step is the computation of the smoothed gradient $\mathbf{X}_u$.

Corollary 2. *If $\mathbf{X}_u$ computed via eigendecomposition then the running time of* SmoothApproxColoring *is $O\left(\epsilon^{-1}n^4\log n\right)$.*

6 The Lovász Shannon-Capacity Problem

The Lovász Shannon-capacity problem (see [22], [14]) on a graph with edge set E can be formulated as the SDP:

$$\begin{array}{l} \max \sum_{i,j} x_{ij}, \\ \text{s.t. } \mathbf{Tr}(\mathbf{X}) + x_{ij} \leq 1, \quad (i,j) \in E \\ \qquad \mathbf{Tr}(\mathbf{X}) - x_{ij} \leq 1, \quad (i,j) \in E \\ \qquad \mathbf{X} \in \{\mathbf{X} \succeq 0 : \mathbf{Tr}(\mathbf{X}) = 1\}, \end{array} \tag{36}$$

This is an SDP with packing constraints. Our method can be used to solve this problem, which, for space reasons we do not include in this extended abstract.

7 Computing the Matrix Exponential

The most expensive step in SmoothApprox and SmoothApproxColoring is SmoothGrad that computes

$$\mathbf{X_u} = \frac{\mathbf{V}\,\mathbf{diag}(e^{n\alpha\boldsymbol{\lambda}})\mathbf{V}^T}{\left(1 + \sum_{i=1}^n e^{n\alpha\lambda_i}\right)} = \frac{e^{n\alpha\mathbf{A}}}{\mathbf{Tr}(e^{n\alpha\mathbf{A}})},$$

where $e^{n\mathbf{A}}$ denotes the *matrix exponential* [23, 24] of

$$\mathbf{A} = \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{L} - 5n\,\mathbf{diag}(\mathbf{u}) \end{bmatrix}.$$

The direct method for computing $e^{n\alpha\mathbf{A}}$ involves computing an eigendecomposition of the matrix $\mathbf{A}$ and then using this to compute $\mathbf{X}_u$ via (22). In this section we discuss a method that computes an ϵ-approximation for $e^{n\alpha\mathbf{A}}$ without first computing the eigendecomposition.

We use a method which we refer to as SI-Lanczos, or the *shift-and-invert* Lanczos, developed by van den Eshof and Hochbruck [31]. The main improvement in [31] results from using Krylov subspaces generated by $(\mathbf{I}+\gamma\mathbf{A})^{-1}$, where $\gamma > 0$ (see also [15]). Therefore, at each iteration, we need to compute a solution to the linear system $\mathbf{y}_{k+1} = (\mathbf{I} + \gamma\mathbf{A})\mathbf{y}_k$. SI-Lanczos works best when $\mathbf{A} \succ 0$, so we approximate the exponential of $\bar{\mathbf{A}} = \mathbf{A} + (6n-2)\mathbf{I}$ (resp. $\bar{\mathbf{A}} = \mathbf{A} + 2n\mathbf{I}$) for the Maxcut (resp. coloring) SDP. This shift ensures that $\bar{\mathbf{A}} \succ 0$ and the condition number $\kappa(\bar{\mathbf{A}}) = \lambda_{\max}(\bar{\mathbf{A}})/\lambda_{\min}(\bar{\mathbf{A}}) \leq 2$. Note that this shift does *not* change the value of $\mathbf{X_u}$ defined in (7). Since the condition number, $\kappa(\bar{\mathbf{A}})$, is bounded, SI-Lanczos can use the well-known conjugate gradient method (see, e.g., Golub and Van Loan [11]) to solve the system of linear equations required at each iteration. The conjugate gradient method computes solutions to linear systems that, at iteration k, have residual error bounded by $[(\sqrt{\kappa(\mathbf{A})} - 1)/(\sqrt{\kappa(\mathbf{A})} + 1)]^k$, which implies the convergence is geometric. Thus, this solves systems of linear equations in $O((n+m)\log(1/\epsilon))$ iterations where $\epsilon > 0$ is the relative error and m is the number of nonzeros in the matrix $\mathbf{A}$. Overall, Theorem 3.3 of [31]

indicates that $O(\log^2(1/\epsilon))$ iterations are required to approximate each column of the exponential. This results an overall complexity of $O(n(n+m)\log^3(1/\epsilon))$. Thus, we have the following corollary.

Corollary 3. *The complexity of computing* $\mathbf{X}_u$ *via* SI-Lanczos *is* $O\big(n(n+m)\log^3(\frac{1}{\epsilon})\big)$. *Therefore, using* SI-Lanczos *for* SmoothGrad *in* SmoothApprox *and* SmoothApproxColoring *results in a complexity of* $O\Big(\epsilon^{-1}n^2(n+m)\log(n)\log^3(\frac{1}{\epsilon})\Big)$.

References

1. F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.
2. S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings, and graph partitionings. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 222–231, 2004.
3. S. J. Benson, Y. Ye, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM J. Optim.*, 10(2):443–461 (electronic), 2000.
4. D. Bienstock. *Potential function methods for approximately solving linear programming problems: theory and practice*. International Series in Operations Research & Management Science, 53. Kluwer Academic Publishers, Boston, MA, 2002.
5. D. Bienstock and G. Iyengar. Solving fractional packing problems in $O^*(\frac{1}{\epsilon})$ iterations. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 146–155, 2004.
6. S. Burer and R. D. C. Monteiro. A projected gradient algorithm for solving the maxcut SDP relaxation. *Optim. Methods Softw.*, 15(3-4):175–200, 2001.
7. L. Fleischer. Fast approximation algorithms for fractional covering problems with box constraint. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms*, 2004.
8. L. K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discrete Math.*, 13(4):505–520 (electronic), 2000.
9. N. Garg and J. Konemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 300–309, 1998.
10. M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.
11. Gene H. Golub and Charles F. Van Loan. *Matrix computations*, volume 3 of *Johns Hopkins Series in the Mathematical Sciences*. Johns Hopkins University Press, Baltimore, MD, 1983.
12. M. D. Grigoriadis and L. G. Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM Journal on Optimization*, 4(1):86–107, February 1994.
13. M. D. Grigoriadis and L. G. Khachiyan. An exponential-function reduction method for block angular convex programs. *Networks*, 26:59–68, 1995.
14. M. Grötschel, L. Lovász, and A. Schrijver. Polynomial algorithms for perfect graphs. In *Topics on perfect graphs*, volume 88 of *North-Holland Math. Stud.*, pages 325–356. North-Holland, Amsterdam, 1984.

15. M. Hochbruck and C. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, 1997.
16. G. Iyengar, D. J. Phillips, and C. Stein. Approximation algorithms for semidefinite packing problems with applications to maxcut and graph coloring. Technical Report TR-2004-06, Computational Optimization Research Center, Columbia University, 2004. Available at http://www.corc.ieor.columbia.edu/reports/techreports/tr-2004-06.pdf.
17. D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. *J. ACM*, 45(2):246–265, 1998.
18. P. Klein and H-I Lu. Efficient approximation algorithms for semidefinite programs arising from MAX CUT and COLORING. In *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 338–347, New York, 1996. ACM.
19. P. Klein and H-I Lu. Space-efficient approximation algorithms for MAXCUT and COLORING semidefinite programs. In *Algorithms and computation (Taejon, 1998)*, volume 1533 of *Lecture Notes in Comput. Sci.*, pages 387–396. Springer, Berlin, 1998.
20. P. Klein, S. A. Plotkin, C. Stein, and É. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, 23(3):466–487, June 1994.
21. T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. *Journal of Computer and System Sciences*, 50:228–243, 1995.
22. L. Lovász. On the Shannon capacity of a graph. *IEEE Trans. Inform. Theory*, 25(1):1–7, 1979.
23. C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Rev.*, 20(4):801–836, 1978.
24. C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.*, 45(1):3–49 (electronic), 2003.
25. Yu. Nesterov. Smooth minimization of nonsmooth functions. Technical report, CORE DP, 2003.
26. Yu. Nesterov and A. Nemirovski. *Interior-point polynomial algorithms in convex programming*, volume 13 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994.
27. S. Plotkin, D. B. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.
28. T. Radzik. Fast deterministic approximation for the multicommodity flow problem. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 486–496, 1995.
29. F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *Journal of the ACM*, 37:318 – 334, 1990.
30. M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling,. *Journal of the ACM*, 48(2):206–242, 2001.
31. J. van den Eshof and M. Hochbruck. Preconditioning lanczos approximations to the matrix exponential, 2004. To appear in SIAM J. of Sci. Comp.

On the Inefficiency of Equilibria in Congestion Games

Extended Abstract

José R. Correa[1], Andreas S. Schulz[2], and Nicolás E. Stier-Moses[3]

[1] School of Business, Universidad Adolfo Ibáñez,
Av. Presidente Errázuriz 3485, Las Condes, Santiago, Chile
correa@uai.cl
[2] Sloan School of Management, Massachusetts Institute of Technology,
Office E53-361, 77 Massachusetts Ave., Cambridge, MA 02139, USA
schulz@mit.edu
[3] Graduate School of Business, Columbia University,
Uris Hall, Room 418, 3022 Broadway Ave., New York, NY 10027, USA
nicolas.stier@columbia.edu

Abstract. We present a short geometric proof for the price of anarchy results that have recently been established in a series of papers on selfish routing in multicommodity flow networks. This novel proof also facilitates two new types of results: On the one hand, we give pseudo-approximation results that depend on the class of allowable cost functions. On the other hand, we derive improved bounds on the inefficiency of Nash equilibria for situations in which the equilibrium travel times are within reasonable limits of the free-flow travel times. These tighter bounds help to explain empirical observations in vehicular traffic networks. Our analysis holds in the more general context of congestion games, which provides the framework in which we describe this work.

1 Introduction

Congestion games (Rosenthal 1973) are noncooperative games in which a player's strategy is to choose a subset of resources, and the utility of each player only depends on the number of players choosing the same or some overlapping strategy. In this paper, we consider a particular variant of *atomic* congestion games, Rosenthal's original setting where the number of players is finite, as well as *nonatomic* congestion games. Nonatomic games (Schmeidler 1973) model interactions involving a large number of players, each with a negligible ability to affect the others. Nonatomic congestion games have been studied, among others, by Milchtaich (2000, 2004), Chau and Sim (2003), and Roughgarden and Tardos (2004).

The most prominent example of a nonatomic congestion game is the traffic routing model of Wardrop (1952). The arcs in a given network represent the resources, the different origin-destination pairs correspond to the player types,

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 167–181, 2005.

and the strategies available to a particular player type are the paths in the network between its origin-destination pair. The cost of an arc describes the delay experienced by traffic traversing that arc as a function of the flow on that arc. A social optimum corresponds to a multicommodity flow of minimum total delay, while a Nash equilibrium equals a user equilibrium flow, where every player is traveling on a shortest path under the prevailing conditions.

Nash equilibria in general and user equilibria in particular are typically inefficient: They generally do not minimize the social cost. Koutsoupias and Papadimitriou (1999) proposed to analyze the inefficiency of equilibria from a worst-case perspective; this led to the notion of "price of anarchy" (Papadimitriou 2001), which is the ratio of the worst social cost of a Nash equilibrium to the cost of an optimal solution. In the context of selfish routing (i.e., the traffic model described in the previous paragraph), the price of anarchy was analyzed in a series of papers for increasingly more general classes of cost functions and model features; see, among others, Roughgarden and Tardos (2002), Roughgarden (2003), Schulz and Stier-Moses (2003), Chau and Sim (2003), Correa, Schulz, and Stier-Moses (2004), Roughgarden and Tardos (2004), Perakis (2004), and Roughgarden (2005).

In this paper, we give alternative proofs for most results in the above-mentioned papers. Our proofs simplify and unify previous arguments, and they enable us to extend these insights to more general settings. In Section 2, we study nonatomic congestion games with separable cost functions. For two known bounds on the inefficiency of Nash equilibria, we provide proofs that rely on a new interpretation of the parameter β introduced by Stier-Moses (2004) in the context of traffic routing. This interpretation sets the stage for various generalizations. In particular, we obtain new tight pseudo-approximation (sometimes also called bicriteria) results that depend on the class of allowable cost functions (Sections 2.1 and 2.2), and a more realistic bound on the price of anarchy for situations where the variable cost of a resource does not exceed its fixed cost by too much (Section 2.3). Section 2.4 considers cost functions that do not include fixed costs. We show that for polynomials of small degree the price of anarchy is significantly smaller than with arbitrary fixed costs. In Sections 3 and 4 we extend these results to atomic congestion games with divisible demands and to nonseparable cost functions, respectively.

2 Nonatomic Congestion Games with Separable Costs

In nonatomic congestion games, we are given a finite set A of resources, and k different types of players. Players are infinitesimal agents, and the continuum of players of type i is represented by the interval $[0, n_i]$. Each player type i possesses a set $\mathcal{S}_i$ of strategies, where each strategy consists of a subset of the resources. For notational convenience, we assume that the sets $\mathcal{S}_i$, $i = 1, 2, \ldots, k$, are disjoint, and we denote their union by $\mathcal{S}$. The rate of consumption of a resource $a \in S$ by a strategy $S \in \mathcal{S}_i$ is given by $r_{a,S} \geq 0$. Each player selects a strategy, which leads to a strategy distribution $x = (x_S)_{S \in \mathcal{S}}$ with $\sum_{S \in \mathcal{S}_i} x_S = n_i$

for each player type i, and $x_S \geq 0$ for all $S \in \mathcal{S}$. A strategy distribution generates a utilization rate $x_a := \sum_{i=1}^{k} \sum_{S \in \mathcal{S}_i : a \in S} r_{a,S}\, x_S$ for each resource $a \in A$. The cost $c_S(x)$ of using a strategy S is the sum of the costs of the resources associated with that strategy, amplified with the corresponding rate of consumption. The cost of resource $a \in A$ is given by a nondecreasing and continuous function $c_a : \mathbb{R}_+^A \to \mathbb{R}_+$. Hence, $c_S(x) := \sum_{a \in S} r_{a,S}\, c_a(x)$. The social cost $C(x)$ of a strategy distribution x is defined as the total disutility experienced by the players: $C(x) := \sum_{i=1}^{k} \sum_{S \in \mathcal{S}_i} c_S(x) x_S = \sum_{a \in A} c_a(x) x_a$.[1]

A social optimum x^{OPT} is a strategy distribution of minimum social cost; i.e., $C(x^{\mathrm{OPT}}) \leq C(x)$ for all strategy distributions x. A strategy distribution x^{NE} is a Nash equilibrium when no player has an incentive to unilaterally change his or her strategy; i.e., $c_S(x^{\mathrm{NE}}) \leq c_{S'}(x^{\mathrm{NE}})$ for any two strategies $S, S' \in \mathcal{S}_i$ with $x_S^{\mathrm{NE}} > 0$, for each $i = 1, 2, \ldots, k$. It is well known (e.g., Dafermos and Sparrow (1969), Smith (1979)) that a strategy distribution x^{NE} is a Nash equilibrium if and only if it satisfies

$$\sum_{a \in A} c_a(x^{\mathrm{NE}})(x_a^{\mathrm{NE}} - x_a) \leq 0 \text{ for all strategy distributions } x. \tag{1}$$

In this section we present different bounds on the inefficiency of equilibria in nonatomic congestion games with separable cost functions. The cost functions c_a are separable if $c_a(x) = c_a(x_a)$ for all $a \in A$. Beckmann, McGuire, and Winsten (1956) showed that in this case a Nash equilibrium x^{NE} exists, and any two different equilibria have the same social cost. While two of the bounds are known, at least in the context of selfish routing, we propose a different way to look at their proofs, which allows us to derive additional types of bounds.

2.1 Affine Cost Functions

Let us begin by studying the simplest case of nonatomic congestion games, namely that of separable and affine cost functions. In other words, the cost of resource $a \in A$ under the utilization rate vector x is $c_a(x) = c_a x_a + b_a$ for some nonnegative coefficients c_a, b_a. The most compact proof of the following result was presented in the context of selfish routing by Correa, Schulz, and Stier-Moses (2004), and our proof below can be viewed as a geometric variant of it. The result itself is due to Roughgarden and Tardos (2002, 2004).

Theorem 1 (Roughgarden and Tardos 2004). *Let x^{NE} be a Nash equilibrium of a nonatomic congestion game with separable, affine cost functions, and let x^{OPT} be a social optimum. Then, $C(x^{\mathrm{NE}}) \leq 4/3\, C(x^{\mathrm{OPT}})$.*

Proof. Let x be an arbitrary strategy distribution. Because of (1), we have

$$C(x^{\mathrm{NE}}) \leq \sum_{a \in A} c_a(x_a^{\mathrm{NE}}) x_a = \sum_{a \in A} c_a(x_a) x_a + \sum_{a \in A} (c_a(x_a^{\mathrm{NE}}) - c_a(x_a)) x_a\,. \tag{2}$$

[1] Because of the equivalence of the strategy-by-strategy view and the resource-by-resource view of the social cost, we interchangeably use x to denote a strategy distribution and its associated vector of utilization rates.

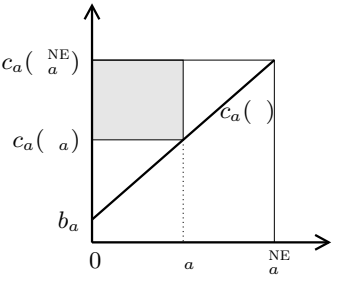


Fig. 1. Illustration of the proof of Theorem 1

Since each function $c_a(\cdot)$ is nondecreasing, we only need to focus on the expressions $(c_a(x_a^{\mathrm{NE}}) - c_a(x_a))x_a$ for which $x_a < x_a^{\mathrm{NE}}$ to bound the last term from above. In this case, $(c_a(x_a^{\mathrm{NE}}) - c_a(x_a))x_a$ is equal to the area of the shaded rectangle in Figure 1. Note that the area of any rectangle whose upper-left corner point is $(0, c_a(x_a^{\mathrm{NE}}))$ and whose lower-right corner point lies on the line representing $c_a(x_a) = c_a x_a + b_a$, is at most half that of the triangle defined by the three points $(0, c_a(x_a^{\mathrm{NE}}))$, $(0, b_a)$, and $(x_a^{\mathrm{NE}}, c_a(x_a^{\mathrm{NE}}))$. In particular,

$$(c_a(x_a^{\mathrm{NE}}) - c_a(x_a))x_a \leq \frac{1}{4}\, c_a(x_a^{\mathrm{NE}})x_a^{\mathrm{NE}} \ ,$$

which completes the proof. □

An immediate consequence of this proof is a pseudo-approximation result, which upper bounds the social cost of a Nash equilibrium by that of an optimal strategy distribution for the same game with more players of each type. More precisely, one only needs the following inequality derived in the preceding proof,

$$\sum_{a \in A} c_a(x_a^{\mathrm{NE}})x_a \leq C(x) + \frac{1}{4} C(x^{\mathrm{NE}}) \ , \tag{3}$$

which holds for *any* nonnegative vector x (i.e., x need not be a strategy distribution), to obtain this result:

Corollary 2. *If x^{NE} is a Nash equilibrium of a nonatomic congestion game with separable, affine cost functions and x^{OPT} is a social optimum for the same game with 5/4 times as many players of each type*[2]*, then $C(x^{\mathrm{NE}}) \leq C(x^{\mathrm{OPT}})$.*

Proof. Let x be an arbitrary strategy distribution of the nonatomic congestion game with 5/4 times as many players as in the original game. Then,

$$C(x^{\mathrm{NE}}) = \frac{5}{4} \sum_{a \in A} c_a(x_a^{\mathrm{NE}})x_a^{\mathrm{NE}} - \frac{1}{4}\, C(x^{\mathrm{NE}}) \leq \sum_{a \in A} c_a(x_a^{\mathrm{NE}})x_a - \frac{1}{4} C(x^{\mathrm{NE}}) \leq C(x) \ .$$

[2] Formally, the continuum of players of type i in the new game is represented by the interval $[0, \frac{5}{4}\ _i]$, $i = 1, 2, \ldots, k$.

The first inequality is implied by (1) and the feasibility of the vector $4\,x/5$ for the original game, while the second one follows from (3). □

The first result of this kind was given by Roughgarden and Tardos (2002), who showed that an equilibrium traffic assignment causes a total travel time of at most that of a social optimum routing twice as much traffic. This result and its subsequent extension to general nonatomic congestion games (Roughgarden and Tardos 2004) hold for arbitrary (separable) cost functions. The selfish routing version of Corollary 2 is due to Chakrabarty (2004) and inspired us to qualify the pseudo-approximation bounds according to the class of cost functions considered; see the next section for details.

2.2 General Cost Functions

Note that we used the linearity of the cost functions $c_a(\cdot)$ in only one place, namely when we proved (3). Hence, a suited generalization of (3) is the key for extending the results in Theorem 1 and Corollary 2 to more general classes of cost functions. For arbitrary (but still separable) cost functions, we are confronted with the situation depicted in Figure 2. To upper bound the area of the shaded rectangle, i.e., $(c_a(x_a^{\mathrm{NE}}) - c_a(x_a))x_a$, in terms of the area of the large rectangle, which is of size $c_a(x_a^{\mathrm{NE}})x_a^{\mathrm{NE}}$, we simply define for each cost function $c_a(\cdot)$ and nonnegative scalar $v_a \geq 0$,

$$\beta(c_a, v_a) := \max_{x_a \geq 0} \frac{(c_a(v_a) - c_a(x_a))x_a}{c_a(v_a)v_a} \quad .$$

Here, $0/0 = 0$ by convention. Note that $0 \leq \beta(c_a, v_a) \leq 1$. For a given class $\mathcal{C}$ of cost functions (e.g., polynomials of a certain degree), we let

$$\beta(\mathcal{C}) := \sup_{c_a \in \mathcal{C}, v_a \geq 0} \beta(c_a, v_a) \quad .$$

This definition leads directly to the following lemma.

Lemma 3. *Let x^{NE} be a Nash equilibrium of a nonatomic congestion game with separable cost functions drawn from a given class $\mathcal{C}$, and let x be an arbitrary*

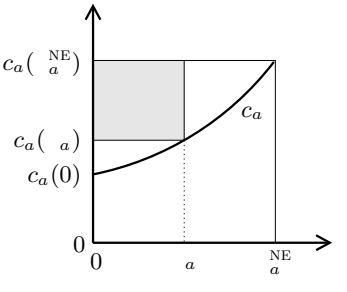


Fig. 2. Illustration of the definition of

nonnegative vector. Then,

$$\sum_{a \in A} c_a(x_a^{\mathrm{NE}}) x_a \leq C(x) + \beta(\mathcal{C}) C(x^{\mathrm{NE}}) \ .$$

In turn, Lemma 3 yields the following generalizations of Theorem 1 and Corollary 2, with virtually no change in proof except for the replacement of (3) with Lemma 3.

Theorem 4. *Let x^{NE} be a Nash equilibrium of a nonatomic congestion game with separable cost functions drawn from a given class $\mathcal{C}$.*

(a) *If x^{OPT} is a social optimum of this game, then $C(x^{\mathrm{NE}}) \leq (1-\beta(\mathcal{C}))^{-1} C(x^{\mathrm{OPT}})$.*
(b) *If x^{OPT} is a social optimum of the same game with $1 + \beta(\mathcal{C})$ times as many players of each type, then $C(x^{\mathrm{NE}}) \leq C(x^{\mathrm{OPT}})$.*

The parameter $\beta(\mathcal{C})$ was first defined in the context of selfish routing by Correa, Schulz, and Stier-Moses (2004), but without mentioning the motivation offered by Figure 2. It is related to the anarchy value $\alpha(\mathcal{C})$ of Roughgarden (2003) (in the context of selfish routing) and Roughgarden and Tardos (2004), but $\alpha(\mathcal{C})$ is only defined for cost functions $c_a(\cdot)$ that are differentiable and for which $c_a(x_a)x_a$ is convex. If $\alpha(\mathcal{C})$ exists, then $\alpha(\mathcal{C}) = (1 - \beta(\mathcal{C}))^{-1}$. Concrete values of $\beta(\mathcal{C})$ can readily be calculated for several classes of cost functions; see Roughgarden (2003) and Correa, Schulz, and Stier-Moses (2004) as well as Figure 4 (a) and Table 1 below.

2.3 Cost Functions with Limited Congestion Effects

To motivate our next set of results, it is helpful to consider selfish routing in traffic networks. The empirically observed ratio of the total travel time of a user equilibrium to that of a system optimum is typically significantly smaller than predicted by the price-of-anarchy results of Theorem 4. For instance, in the computational studies of Jahn, Möhring, Schulz, and Stier-Moses (2005) the largest ratio of user equilibrium cost to system optimum cost over several realistic instances is 1.15 (instead of the theoretical worst case of 2.151). Qiu et al. (2003) made a similar observation in the context of telecommunication networks. For a given class of latency functions, the corresponding price of anarchy is a worst-case measure, taken over all possible instances. However, if one compares the time needed to drive to work during rush hour with the duration of the same trip at night, then the "free-flow travel time" is usually not a negligible fraction of the rush-hour experience. Formally, we make the following assumption: The cost of any given resource at utilization rate zero is at least a constant fraction of that of the same resource at the utilization rate in equilibrium. A different way of illustrating this assumption is by looking at a production process, where the fixed cost of any resource accounts for a substantial fraction of the total cost (fixed plus variable costs) in equilibrium. Depending on the ratio of fixed to total cost, we obtain a parametrized sequence of improved bounds on the inefficiency of equilibria.

Theorem 5. *Let x^{NE} be a Nash equilibrium of a nonatomic congestion game with separable cost functions drawn from a given class $\mathcal{C}$ such that $c_a(0) \geq \eta\, c_a(x_a^{\mathrm{NE}})$ for all $a \in A$, for some constant $0 \leq \eta \leq 1$.*[3]

(a) *If x^{OPT} is a social optimum, then $C(x^{\mathrm{NE}}) \leq (1-(1-\eta)\beta(\mathcal{C}))^{-1}\, C(x^{\mathrm{OPT}})$.*
(b) *If x^{OPT} is a social optimum of the same game with $1+(1-\eta)\beta(\mathcal{C})$ times as many players of each type, then $C(x^{\mathrm{NE}}) \leq C(x^{\mathrm{OPT}})$.*

The proof of Theorem 5 rests upon the following generalization of Lemma 3.

Lemma 6. *Let x^{NE} be a Nash equilibrium of a nonatomic congestion game with separable cost functions drawn from a given class $\mathcal{C}$ such that $c_a(0) \geq \eta\, c_a(x_a^{\mathrm{NE}})$ for all $a \in A$, for some constant $0 \leq \eta \leq 1$. Moreover, let x be an arbitrary nonnegative vector. Then,*

$$\sum_{a \in A} c_a(x_a^{\mathrm{NE}}) x_a \leq C(x) + (1-\eta)\beta(\mathcal{C}) C(x^{\mathrm{NE}}) \ .$$

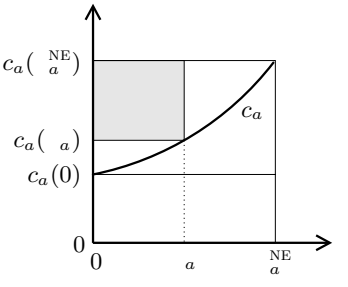


Fig. 3. Illustration of the proof of Lemma 6

Proof. We again rewrite $c_a(x_a^{\mathrm{NE}})x_a$ as $c_a(x_a)x_a + (c_a(x_a^{\mathrm{NE}}) - c_a(x_a))x_a$. If $x_a \geq x_a^{\mathrm{NE}}$, there is nothing left to prove. So assume $x_a < x_a^{\mathrm{NE}}$, and consider Figure 3. Because $c_a(0) \geq \eta\, c_a(x_a^{\mathrm{NE}})$, the area $(c_a(x_a^{\mathrm{NE}}) - c_a(x_a))x_a$ of the small shaded rectangle is at most $\beta(\mathcal{C})$ times that of the rectangle with upper-left corner point $(0, c_a(x_a^{\mathrm{NE}}))$ and lower-right corner point $(x_a^{\mathrm{NE}}, c_a(0))$, which is of size at most $(1-\eta)c_a(x_a^{\mathrm{NE}})x_a^{\mathrm{NE}}$. The result follows. □

Figure 4 (a) displays the relationship between η and the bound on the price of anarchy given by Theorem 5 (a) for polynomials of different degrees. Note that the price of anarchy is at most $1/\eta$, even if we do not place any restriction on $\mathcal{C}$. This observation qualifies the unboundedness of the price of anarchy for instances

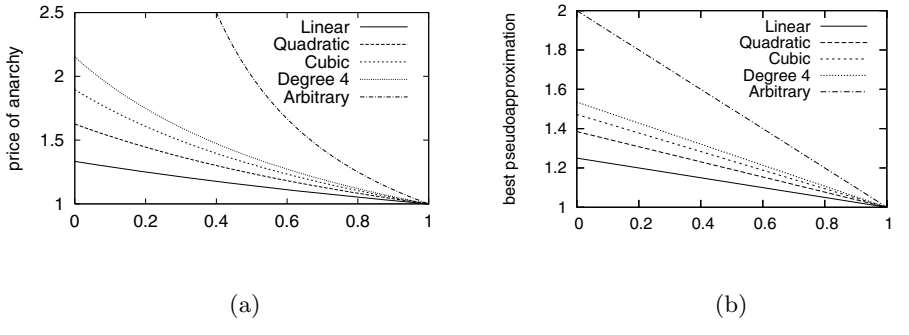
[3] We also assume that $\mathcal{C}$ is closed under adding constants $b \in \mathbb{R}$ for which $c(0)+b \geq 0$; otherwise, the resulting improvement would not be directly visible but would remain hidden in the value of $\beta(\mathcal{C})$.

with general cost functions described by Roughgarden and Tardos (2002, 2004). As another example, consider a vehicular network in which users travel at most twice as long when the network is congested compared to the situation when it is not. If arc latencies are modeled as polynomials of degree 4, as it is the case for the widely used Bureau of Public Roads' functions (1964), Theorem 4 (a) gives an upper bound of 2.151 on the price of anarchy. However, Theorem 5 (a) gives a more accurate bound of 1.365. More generally, we believe that the bounds presented in Theorem 5 offer a good explanation of the satisfactory performance of Nash equilibria in many practical situations. Figure 4 (b) illustrates part (b) of Theorem 5.

Let us also point out that the bounds given in Theorem 5 are tight. Consider the traffic assignment instance in Figure 5, where v units of flow must be routed from one node to the other over two parallel arcs a and a'. The arc latencies are $c_a(x_a) = c(v)$ (a constant) and $c_{a'}(x_{a'}) = \eta\, c(v) + (1-\eta)c(x_{a'})$, respectively. Here, the function c and the scalar v are chosen such that $\beta(\mathcal{C}) = \beta(c, v)$. Both bounds given in Theorem 5 are simultaneously tight for this instance.

2.4 Cost Functions Without Fixed Costs

In contrast to the preceding section, we now consider instances where the cost of each resource at utilization rate zero is equal to zero. This model helps to capture situations in which variable costs dominate fixed costs or where fixed costs can be neglected altogether. Examples include telecommunication networks where the propagation delays are usually negligible compared to the queueing delays in the routers, and production systems where the equipment has been installed already. We give the first results of this kind; in particular, the price of anarchy for small-degree polynomials in this setting is smaller than the bounds given in Theorem 4. Polynomials are of particular interest in several application contexts. For instance, the Bureau of Public Roads' cost functions used in traffic networks are polynomials of degree 4.



(a) (b)

Fig. 4. (a) Price of anarchy as a function of fixed to total costs. (b) Minimum value for which the pseudo-approximation result holds. Each curve refers to a set $\mathcal{C}$ that contains nonnegative polynomials of fixed degree

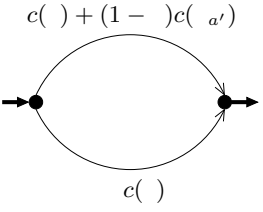


Fig. 5. Instance for which the bounds in Theorem 5 are tight

Dafermos and Sparrow (1969) showed that when all cost functions are monomials of the same degree, Nash equilibria and system optima coincide. In particular, the price of anarchy is 1 if we only allow linear functions (affine functions without constant term) instead of 4/3, the value for the set of all affine functions. Before we show that a similar effect can be observed for polynomials of (somewhat) higher degree, let us introduce some notation to facilitate the proofs. We represent the cost functions as $c_a(x_a) := \sum_{1 \le i \le d} c_{a,i}\, x_a^i$ whenever we consider polynomials of degree d. Moreover, we denote the total cost of a strategy distribution x when resources are utilized according to another vector x' by $C^{x'}(x) := \sum_{a \in A} c_a(x'_a) x_a$. Also, we denote the total cost restricted to the degree i monomials by $C_i(x) := \sum_{a \in A} c_{a,i}\, x_a^{i+1}$ and $C_i^{x'}(x) := \sum_{a \in A} c_{a,i}\, x_a\, (x'_a)^i$, respectively. Using this notation, we can rewrite (1) as $C(x^{\mathrm{NE}}) \le C^{x^{\mathrm{NE}}}(x)$, which is valid for any strategy distribution x and Nash equilibrium x^{NE}. The following lemma is well known in the context of variational inequalities; see, e.g., Smith (1979).

Lemma 7. *If x^{NE} is a Nash equilibrium and x is an arbitrary strategy distribution of a nonatomic congestion game, then $C^x(x^{\mathrm{NE}}) \le C(x)$.*

With the notation and preliminary results in place, we are prepared to bound the price of anarchy in networks without fixed costs.

Theorem 8. *Consider the family of nonatomic congestion games with separable cost functions that are polynomials of degree at most d, for $1 \le d \le 4$. Then, the price of anarchy for this class of games is bounded from above and below by the values displayed in Table 1.*

Due to space limitations, we restrict ourselves to prove the upper bound of 5/4 for cubic functions.

Proof. We use two simple bounds: $(\delta x_a - x_a^{\mathrm{NE}})^2 \ge 0$ with $\delta \in \mathbb{R}$, which implies that $C_1^x(x^{\mathrm{NE}}) \le C_1(x^{\mathrm{NE}}) + \frac{1}{4} C_1(x)$ and $C_2^{x^{\mathrm{NE}}}(x) \le \frac{1}{2} C_2^x(x^{\mathrm{NE}}) + \frac{1}{2} C_2(x^{\mathrm{NE}})$, and $(\frac{1}{2} x_a^2 - (x_a^{\mathrm{NE}})^2 + x_a\, x_a^{\mathrm{NE}})^2 \ge 0$, which implies that $C_3^{x^{\mathrm{NE}}}(x) \le \frac{1}{2} C_3^x(x^{\mathrm{NE}}) + \frac{1}{2} C_3(x^{\mathrm{NE}}) + \frac{1}{8} C_3(x)$. Now,

$$C(x^{\mathrm{NE}}) \le C^{x^{\mathrm{NE}}}(x) = C_3^{x^{\mathrm{NE}}}(x) + C_2^{x^{\mathrm{NE}}}(x) + C_1^{x^{\mathrm{NE}}}(x)$$

$$\leq \frac{C_3^x(x^{\mathrm{NE}})}{2} + \frac{C_3(x^{\mathrm{NE}})}{2} + \frac{C_3(x)}{8} + \frac{C_2^x(x^{\mathrm{NE}})}{2} + \frac{C_2(x^{\mathrm{NE}})}{2} + \frac{C_1^x(x^{\mathrm{NE}})}{2} + \frac{C_1(x^{\mathrm{NE}})}{2} + \frac{C_1(x)}{8}$$

$$\leq \frac{C(x^{\mathrm{NE}})}{2} + \frac{C^x(x^{\mathrm{NE}})}{2} + \frac{C(x)}{8} \leq \frac{1}{2}C(x^{\mathrm{NE}}) + \frac{5}{8}C(x) \ .$$

□

Table 1. Comparison of guarantees for the efficiency of equilibria with and without fixed costs. All coefficients are assumed to be nonnegative. The column entitled '$c_a(0) = 0$' shows lower and upper bounds on the price of anarchy without fixed costs. The lower bounds represent the worst-case coordination ratio of a Pigou-type instance with two parallel arcs, cost functions in $\mathcal{C}$, and no fixed costs. The column entitled '$c_a(0)$ arbitrary' shows the exact price of anarchy with arbitrary fixed costs

Set $\mathcal{C}$ of allowable cost functions	Example	Price of Anarchy ($\mathcal{C}$)		
		$c_a(0) = 0$		$c_a(0)$ arbitrary
		LB	UB	
linear functions	$a_1 \ + a_0$	1	1	1.334
quadratic functions	$a_2 \ ^2 + a_1 \ + a_0$	1.035	1.185	1.626
cubic functions	$a_3 \ ^3 + a_2 \ ^2 + a_1 \ + a_0$	1.098	1.25	1.896
polynomials of degree 4	$\sum_{i=0}^{4} a_i \ ^i$	1.167	1.999	2.151

3 Weighted Congestion Games with Divisible Demands

Rosenthal's (1973) original version of congestion games has a finite number of players, and each player controls one unit of demand that cannot be split across strategies. While he showed that a pure-strategy Nash equilibrium always exists, this is not necessarily the case for weighted congestion games, where players can have arbitrary demands (Fotakis et al. 2004). We consider a hybrid between weighted congestion games and nonatomic congestion games, which is guaranteed to possess a pure-strategy Nash equilibrium. Moreover, the price of anarchy can be bounded in similar terms to the results discussed above.

An *atomic weighted congestion game with divisible demands* is similar to a nonatomic congestion game, except that there is a finite number of players $1, 2, \ldots, k$. Each player i has a divisible demand $n_i > 0$ and assigns portions $x_S \geq 0$ of it to various strategies S in his or her strategy space $\mathcal{S}_i$ such that $\sum_{S \in \mathcal{S}_i} x_S = n_i$. For a given strategy distribution $x = (x_S)_{S \in \mathcal{S}}$, the cost to player i is $C_i(x) := \sum_{S \in \mathcal{S}_i} c_S(x) x_S$. The social cost $C(x)$ is equal to the total cost; i.e., $C(x) := \sum_{i=1}^{k} C_i(x) = \sum_{a \in A} c_a(x_a) x_a$, where $x_a = \sum_{i=1}^{k} \sum_{a \in S \in \mathcal{S}_i} r_{a,S} x_S$ is the utilization rate of resource a. A strategy distribution x constitutes a Nash equilibrium if, for each player i, $(x_S)_{S \in \mathcal{S}_i}$ minimizes $C_i(x)$, keeping the actions of the other players fixed. This model provides an abstract framework for the "atomic splittable selfish routing problem" considered

in previous papers. Catoni and Pallotino (1991) presented instances for which the ratio of the equilibrium cost to the optimal cost is smaller for the nonatomic version than its atomic splittable counterpart, while Roughgarden (2005) proved that the price of anarchy for any given class of allowable cost functions is always dominated by that of nonatomic games. Harker (1998) and Orda, Rom, and Shimkin (1993) established the existence of pure-strategy Nash equilibria, and Roughgarden and Tardos (2002) showed that their pseudo-approximation results for nonatomic selfish routing games also holds for the atomic splittable case.

Although the variational inequality (1) does in general not hold for Nash equilibria of atomic games with divisible demands, it actually suffices to prove that (2) is still true in order to derive results similar to Theorems 4 and 5. Consider a Nash equilibrium x^{NE} and define, for $a \in A$, $\bar{c}_a(x_a) := c_a(x_a^{\mathrm{NE}})$ if $x_a \le x_a^{\mathrm{NE}}$ and $\bar{c}_a(x_a) := c_a(x_a)$, otherwise. It is straightforward to extend a result in the proof of Roughgarden and Tardos (2002, Theorem 5.4) to characterize Nash equilibria in atomic congestion games with divisible demands:

Lemma 9 (Roughgarden and Tardos 2002). *Let x^{NE} be a Nash equilibrium of an atomic congestion game with divisible demands. Furthermore, assume that the functions $c_a(x_a)x_a$ are convex, for all $a \in A$. Then, $\sum_{a\in A} \bar{c}_a(x_a^{\mathrm{NE}})x_a^{\mathrm{NE}} \le \sum_{a\in A} \bar{c}_a(x_a)x_a$ for all strategy distributions x.*

Using slightly weaker versions of (2) and Lemma 3, we can bound the price of anarchy of atomic congestion games with divisible demands.

Theorem 10. *Let x^{NE} be a Nash equilibrium of an atomic congestion game with divisible demand and separable cost functions drawn from a given class $\mathcal{C}$. Assume that $c_a(x_a)x_a$ is a convex function, for all $c_a \in \mathcal{C}$.*

(a) *If x^{OPT} is a social optimum of this game, then $C(x^{\mathrm{NE}}) \le (1-\beta(\mathcal{C}))^{-1}\, C(x^{\mathrm{OPT}})$.*
(b) *If x^{OPT} is a social optimum of the same game with $1+\beta(\mathcal{C})$ times as many players of each type, then $C(x^{\mathrm{NE}}) \le C(x^{\mathrm{OPT}})$.*

Proof. Lemma 9 implies that the cost $C(x^{\mathrm{NE}})$ of a Nash equilibrium is bounded from above by $\sum_{a\in A} \bar{c}_a(x_a)x_a$ for any strategy distribution x. Let us consider the terms $\bar{c}_a(x_a)x_a$ individually. If $x_a \ge x_a^{\mathrm{NE}}$, then $\bar{c}_a(x_a)x_a = c_a(x_a)x_a$, and we are done. Otherwise, $\bar{c}_a(x_a)x_a = c_a(x_a)x_a + (c_a(x_a^{\mathrm{NE}}) - c_a(x_a))x_a$. Hence, $C(x^{\mathrm{NE}}) \le \sum_{a\in A} \bar{c}_a(x_a)x_a \le C(x) + \beta(\mathcal{C})C(x^{\mathrm{NE}})$, which shows part (a). For part (b), notice that $(1+\beta(\mathcal{C}))^{-1}x^{\mathrm{OPT}}$ is feasible for the original problem. Therefore,

$$C(x^{\mathrm{NE}}) \le \sum_{a\in A} \bar{c}_a\left(\frac{x_a^{\mathrm{OPT}}}{1+\beta(\mathcal{C})}\right)\frac{x_a^{\mathrm{OPT}}}{1+\beta(\mathcal{C})} \le \frac{1}{1+\beta(\mathcal{C})}\sum_{a\in A} \bar{c}_a(x_a^{\mathrm{OPT}})x_a^{\mathrm{OPT}},$$

where the second inequality follows from the monotonicity of the cost functions. Now, we can proceed as in Corollary 2:

$$C(x^{\mathrm{NE}}) = (1+\beta(\mathcal{C}))C(x^{\mathrm{NE}}) - \beta(\mathcal{C})\, C(x^{\mathrm{NE}}) \le \sum_{a\in A} \bar{c}_a(x_a^{\mathrm{OPT}})x_a^{\mathrm{OPT}} - \beta(\mathcal{C})\, C(x^{\mathrm{NE}}) \le C(x)\,.$$

□

Part (a) of Theorem 10 extends and simplifies a result of Roughgarden (2005) who proved a similar bound for network games. Part (b) is new. Using the arguments in the proof of Theorem 10, it is straightforward to extend Theorem 5 to atomic games.

4 Nonatomic Games with Nonseparable Costs

In some practical situations, the cost of using one resource may depend on the rate of consumption of others. For instance, the time a vehicle needs to cross through a stop sign clearly depends on the amount of flow traversing the perpendicular street; the waiting time of passengers at a given bus stop depends on the number of passengers boarding the bus at previous stops; or, to give an example in the context of wireless communication networks, transmission delays depend on the load of neighboring cells, because of interference.

The purpose of this section is to extend our results on nonatomic congestion games to the case where cost functions are not necessarily separable. In this context it is convenient to write the social cost of a strategy distribution x as an inner product, $C(x) = \langle c(x), x\rangle$, with $c : X \to \mathbb{R}_+^A$ continuous. Here, X denotes the convex and compact space of feasible utilization vectors. (In particular, X can be used to model side constraints such as resource capacities; see Correa, Schulz, and Stier-Moses (2004) for a discussion.) In the spirit of Equation (1), a strategy distribution x^{NE} is an equilibrium if it satisfies the variational inequality

$$\langle c(x^{\mathrm{NE}}), x^{\mathrm{NE}} - x\rangle \le 0 \quad \text{for all } x \in X. \tag{4}$$

Moreover, a social optimum x^{OPT} is an optimal solution to $\min_{x\in X}\langle c(x), x\rangle$. Under the continuity of c as well as the compactness and convexity of X, an equilibrium exists by the classic result of Hartman and Stampacchia (1966) (see also Smith 1979). Of course, a system optimum also exists as c is continuous and X is compact.

For a cost function c and a utilization vector $v \in \mathbb{R}_+^A$, a natural extension of the parameter β is

$$\beta(c, v) := \max_{x\ge 0} \frac{\langle c(v) - c(x), x\rangle}{\langle c(v), v\rangle} \quad .$$

With the definition of $\beta(\mathcal{C}) := \sup_{c\in\mathcal{C}, v\in X} \beta(c, v)$, we can extend Lemma 3 to nonseparable cost functions, thereby simplifying and extending earlier work. Chau and Sim (2003) proved that the price of anarchy for nonseparable and symmetric cost functions is bounded by a natural extension of the parameter $\alpha(\mathcal{C})$ of Roughgarden and Tardos (see Section 2.2). Perakis (2004) considered general nonseparable cost functions. Let us mention that the known bounds require stronger assumptions on the cost functions, such as convexity, differentiability, and monotonicity.

Lemma 11. *Let x^{NE} be an equilibrium of a nonatomic congestion game with cost functions drawn from a class $\mathcal{C}$ of nonseparable cost functions, and let x be a nonnegative vector. Then, $\langle c(x^{\mathrm{NE}}), x\rangle \le C(x) + \beta(\mathcal{C})C(x^{\mathrm{NE}})$.*

This lemma yields the following price-of-anarchy and pseudo-approximation results for nonatomic congestion games with nonseparable cost functions.

Theorem 12. *Let x^{NE} be an equilibrium of a nonatomic congestion game with cost functions drawn from a class $\mathcal{C}$ of nonseparable cost functions.*

(a) *If x^{OPT} is a social optimum for this game, then $C(x^{\mathrm{NE}}) \le (1-\beta(\mathcal{C}))^{-1} C(x^{\mathrm{OPT}})$.*
(b) *If x^{OPT} is a social optimum for the same game with $1+\beta(\mathcal{C})$ times as many players of each type, then $C(x^{\mathrm{NE}}) \le C(x^{\mathrm{OPT}})$.*

Proof. For (a), it suffices to use (4) and Lemma 11:

$$C(x^{\mathrm{NE}}) = \langle c(x^{\mathrm{NE}}), x^{\mathrm{NE}} \rangle \le \langle c(x^{\mathrm{NE}}), x^{\mathrm{OPT}} \rangle \le C(x^{\mathrm{OPT}}) + \beta(\mathcal{C}) C(x^{\mathrm{NE}}) \ .$$

Let us now prove part (b). Because of the feasibility of $(1+\beta(\mathcal{C}))^{-1}\, x^{\mathrm{OPT}}$ for the original game, we have that $\langle c(x^{\mathrm{NE}}), x^{\mathrm{NE}} \rangle \le \langle c(x^{\mathrm{NE}}), (1+\beta(\mathcal{C}))^{-1}\, x^{\mathrm{OPT}} \rangle$. Therefore,

$$\begin{aligned}
\langle c(x^{\mathrm{NE}}), x^{\mathrm{NE}} \rangle &= (1+\beta(\mathcal{C})) \langle c(x^{\mathrm{NE}}), x^{\mathrm{NE}} \rangle - \beta(\mathcal{C}) \langle c(x^{\mathrm{NE}}), x^{\mathrm{NE}} \rangle \\
&\le (1+\beta(\mathcal{C})) \langle c(x^{\mathrm{NE}}), (1+\beta(\mathcal{C}))^{-1} x^{\mathrm{OPT}} \rangle - \beta(\mathcal{C}) \langle c(x^{\mathrm{NE}}), x^{\mathrm{NE}} \rangle \\
&\le C(x^{\mathrm{OPT}}) + \beta(\mathcal{C}) C(x^{\mathrm{NE}}) - \beta(\mathcal{C}) C(x^{\mathrm{NE}}) \\
&= C(x^{\mathrm{OPT}}) \ .
\end{aligned}$$

□

A particular class of nonseparable cost functions that has been studied before are affine functions; i.e., $c(x) = Ax + b$, with $b \ge 0$, and A symmetric and positive semidefinite. Theorem 12 provides a simple proof of a result by Chau and Sim (2003), which established that the price of anarchy for this kind of cost functions is at most 4/3. Indeed, in this case

$$\beta(c, v) = \max_{x \ge 0} \frac{\langle c(v) - c(x), x \rangle}{\langle c(v), v \rangle} = \frac{\max_{x \ge 0} \langle A(v-x), x \rangle}{\langle Av, v \rangle + \langle b, v \rangle} \ .$$

As A is symmetric and positive semidefinite, the numerator amounts to a convex minimization problem, and the optimum is attained at $x = v/2$, leading to $\beta(\text{affine costs}) = 1/4$. Theorem 12 yields $C(x^{\mathrm{NE}}) \le 4/3\, C(x^{\mathrm{OPT}})$, where x^{OPT} is a social optimum for this game. Moreover, $C(x^{\mathrm{NE}}) \le C(x^{\mathrm{OPT}})$, for a social optimum x^{OPT} of the same game with 5/4 times as many players of each type.

Let us finally note that the improved results for games with limited congestion (i.e., Theorem 5) also hold in this setting. Indeed, we only need to generalize Lemma 6. Similarly to Section 2.3, we assume that $\mathcal{C}$ satisfies that for $c \in \mathcal{C}$ and a constant vector $b \in \mathbb{R}^A$ for which $c(x) + b \in \mathbb{R}^A_+$ for all $x \in \mathbb{R}^A_+$, $c(x) + b \in \mathcal{C}$.

Lemma 13. *Let x^{NE} be an equilibrium of a nonatomic congestion game with cost functions drawn from a class $\mathcal{C}$ such that $c(0) \ge \eta\, c(x^{\mathrm{NE}})$ for $c \in \mathcal{C}$ and $0 \le \eta \le 1$. If x is a nonnegative vector, then $\langle c(x^{\mathrm{NE}}), x \rangle \le C(x) + (1-\eta)\beta(\mathcal{C}) C(x^{\mathrm{NE}})$.*

Proof. Let us write $c \in \mathcal{C}$ as $c(x) = M(x) + b$, where $b = c(0) \geq \eta\, c(x^{\mathrm{NE}}) \geq 0$. Thus,

$$\begin{aligned}\beta(c, x^{\mathrm{NE}}) &= \max_{x \geq 0} \frac{\langle c(x^{\mathrm{NE}}) - c(x), x\rangle}{\langle c(x^{\mathrm{NE}}), x^{\mathrm{NE}}\rangle} \leq \max_{x \geq 0} \frac{\langle M(x^{\mathrm{NE}}) - M(x), x\rangle}{\langle M(x^{\mathrm{NE}}), x^{\mathrm{NE}}\rangle + \frac{\eta}{1-\eta}\langle M(x^{\mathrm{NE}}), x^{\mathrm{NE}}\rangle} \\ &= (1-\eta) \max_{x \geq 0} \frac{\langle M(x^{\mathrm{NE}}) - M(x), x\rangle}{\langle M(x^{\mathrm{NE}}), x^{\mathrm{NE}}\rangle} \leq (1-\eta)\beta(\mathcal{C}) \ .\end{aligned}$$

So, $\langle c(x^{\mathrm{NE}}), x\rangle \leq \langle c(x), x\rangle + \beta(c, x^{\mathrm{NE}})\langle c(x^{\mathrm{NE}}), x^{\mathrm{NE}}\rangle \leq C(x) + (1-\eta)\beta(\mathcal{C})C(x^{\mathrm{NE}})$. □

References

Beckmann, M. J., C. B. McGuire, and C. B. Winsten (1956). *Studies in the Economics of Transportation*. Yale University Press, New Haven, CT.

Bureau of Public Roads (1964). Traffic assignment manual. U.S. Department of Commerce, Urban Planning Division, Washington, DC.

Catoni, S. and S. Pallotino (1991). Traffic equilibrium paradoxes. *Transportation Science 25*, 240–244.

Chakrabarty, D. (2004). Improved bicriteria results for the selfish routing problem. Manuscript.

Chau, C. K. and K. M. Sim (2003). The price of anarchy for non-atomic congestion games with symmetric cost maps and elastic demands. *Operations Research Letters 31*, 327–334.

Correa, J. R., A. S. Schulz, and N. E. Stier-Moses (2004). Selfish routing in capacitated networks. *Mathematics of Operations Research 29*, 961–976.

Dafermos, S. C. and F. T. Sparrow (1969). The traffic assignment problem for a general network. *Journal of Research of the U.S. National Bureau of Standards 73B*, 91–118.

Fotakis, D., S. C. Kontogiannis, and P. G. Spirakis (2004). Selfish unsplittable flows. In J. Diaz, J. Karhumäki, A. Lepistö, and D. Sannella (Eds.), *Automata, Languages and Programming: Proceedings of the 31st International Colloquium (ICALP)*, Turku, Finland, Volume 3142 of *Lecture Notes in Computer Science*, pp. 593–605. Springer, Heidelberg.

Harker, P. T. (1988). Multiple equilibrium behaviors of networks. *Transportation Science 22*, 39–46.

Hartman, G. and G. Stampacchia (1966). On some nonlinear elliptic differential equations. *Acta Mathematica 115*, 271–310.

Jahn, O., R. H. Möhring, A. S. Schulz, and N. E. Stier-Moses (2005). System-optimal routing of traffic flows with user constraints in networks with congestion. *Operations Research*. To appear.

Koutsoupias, E. and C. H. Papadimitriou (1999). Worst-case equilibria. In C. Meinel and S. Tison (Eds.), *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, Trier, Germany, Volume 1563 of *Lecture Notes in Computer Science*, pp. 404–413. Springer, Heidelberg.

Milchtaich, I. (2000). Generic uniqueness of equilibrium in large crowding games. *Mathematics of Operations Research 25*, 349–364.

Milchtaich, I. (2004). Social optimality and cooperation in nonatomic congestion games. *Journal of Economic Theory 114*, 56–87.

Orda, A., R. Rom, and N. Shimkin (1993). Competitive routing in multiuser communication networks. *IEEE/ACM Transactions on Networking 1*, 510–521.

Papadimitriou, C. H. (2001). Algorithms, games, and the Internet. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, Hersonissos, Greece, pp. 749–753. ACM Press, New York, NY.

Perakis, G. (2004). The "price of anarchy" under nonlinear and asymmetric costs. In D. Bienstock and G. Nemhauser (Eds.), *Proceedings of the 10th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, New York, NY, Volume 3064 of *Lecture Notes in Computer Science*, pp. 46–58. Springer, Heidelberg.

Qiu, L., Y. R. Yang, Y. Zhang, and S. Shenker (2003). On selfish routing in Internet-like environments. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, Karlsruhe, Germany, pp. 151–162. ACM Press, New York, NY.

Rosenthal, R. W. (1973). A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory 2*, 65–67.

Roughgarden, T. (2003). The price of anarchy is independent of the network topology. *Journal of Computer and System Sciences 67*, 341–364.

Roughgarden, T. (2005). Selfish routing with atomic players. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Vancouver, Canada, pp. 973–974. SIAM, Philadelphia, PA.

Roughgarden, T. and É. Tardos (2002). How bad is selfish routing? *Journal of the ACM 49*, 236–259.

Roughgarden, T. and É. Tardos (2004). Bounding the inefficiency of equilibria in nonatomic congestion games. *Games and Economic Behavior 47*, 389–403.

Schmeidler, D. (1973). Equilibrium points of nonatomic games. *Journal of Statistical Physics 7*, 295–300.

Schulz, A. S. and N. E. Stier-Moses (2003). On the performance of user equilibria in traffic networks. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Baltimore, MD, pp. 86–87. SIAM, Philadelphia, PA.

Smith, M. J. (1979). The existence, uniqueness and stability of traffic equilibria. *Transportation Research 13B*, 295–304.

Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers, Part II, Vol. 1*, 325–378.

Unrelated Parallel Machine Scheduling with Resource Dependent Processing Times*

Alexander Grigoriev[1], Maxim Sviridenko[2], and Marc Uetz[1]

[1] Maastricht University, Quantitative Economics, P.O.Box 616, 6200 MD Maastricht, The Netherlands
{a.grigoriev, m.uetz}@ke.unimaas.nl
[2] IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA
sviri@us.ibm.com

Abstract. We consider unrelated parallel machine scheduling problems with the objective to minimize the schedule makespan. In addition to its machine-dependence, the processing time of any job is also dependent on the usage of a scarce renewable resource. An amount of k units of that resource, e.g. workers, can be distributed over the jobs in process, and the more of that resource is allocated to a job, the smaller its processing time. The model generalizes the classical unrelated machine scheduling problem, adding a resource-time tradeoff. It is also a natural variant of a generalized assignment problem studied previously by Shmoys and Tardos, the difference lying in the fact the resource is renewable and not a total budget constraint. We use a two-phased LP rounding technique to assign resources to jobs and jobs to machines. Combined with Graham's list scheduling, we thus prove the existence of a $(4+2\sqrt{2})$-approximation algorithm. We show how our approach can be adapted to scheduling problems with dedicated machines as well, with an improvement of the performance bound to $(3+2\sqrt{2})$. Moreover, we derive a lower bound of 2 for the employed LP-based analysis, and we prove a (3/2)-inapproximability result.

1 Introduction and Related Work

Unrelated parallel machine scheduling to minimize the makespan, $R||C_{\max}$ in the three-field notation of Graham et al. [6], is one of the classical problems in combinatorial optimization. Given are n jobs that have to be scheduled on m parallel machines, and the processing time of job j on machine i is p_{ij}. The goal is to minimize the latest job completion, the makespan $C_{\max}$. If the number of machines m is not fixed, the best approximation algorithm to date is a

* This work was done while the second author was visiting Maastricht University, partially supported by METEOR, the Maastricht Research School of Economics of Technology and Organizations.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 182–195, 2005.

2-approximation by Lenstra, Shmoys and Tardos [10]. Moreover, the problem cannot be approximated within a factor smaller than 3/2, unless P=NP [10].

Shmoys and Tardos [12] consider the same problem with the additional feature of costs λ_{ij} if job j is processed on machine i. They show that, if a schedule with total cost Λ and makespan T exists, a schedule with total cost Λ and makespan at most $2T$ can be found in polynomial time. The proof relies on rounding the solution of an LP relaxation. They obtain the same result even for a more general version of the problem, namely when the processing time p_{ij} of any job-machine pair is not fixed, but may be reduced linearly, in turn for a linear increase of the associated cost λ_{ij} [12]. Note that, in both versions of the problem studied in [12], the costs λ_{ij} are *non-renewable* resources, such as a monetary budget, with a global budget Λ.

In this paper, we consider a different variant of the problem of [12]. Namely, the processing times p_{ij} of any job-machine pair can be reduced by utilizing a *renewable* resource, such as additional workers, that can be allocated to the jobs. In other words, a maximum number of k units of a resource may be used to speed up the jobs, and the available amount of k units of that resource must not be exceeded at any time. In contrast to the linearity assumption on the costs and processing times in [12], the only assumption we make in this paper is that the processing times p_{ijs}, which now depend also on the number s of allocated resources, are non-increasing in s for each job-machine pair. That is, we assume that $p_{ij0} \geq \cdots \geq p_{ijk}$ for all jobs j and all machines i. The practical motivation to study this problem is evident; to give an example, one may think of production planning where additional (overtime) workers can be allocated to specific tasks within the production in order to reduce the production cycle time.

Related Work. In a manuscript by Grigoriev et al. [7], a restricted version of the problem is addressed. They consider *dedicated*, parallel machines, thus each job is dedicated beforehand to be processed on a given machine. Moreover, their model is restricted to a binary resource, thus the availability of the additional resource is $k = 1$. Any job may be processed either with or without using that resource, with a reduced processing time if the resource is used. Finally, the number of machines m in their paper is considered fixed, and not part of the input. For that problem, they derive a $(3+\varepsilon)$–approximation, and for the problem with $m = 2$ machines, they derive (weak) NP-hardness and a fully polynomial time approximation scheme [7].

Jobs with resource dependent processing times also appear in the literature as *malleable* or *parallelizable tasks*, e.g. in [11, 13]. In these models, jobs can be processed on one or more parallel processors, and they have non-increasing processing times p_{js} in the number s of processors used. Any processor can only handle one job at a time, and the goal is to minimize the schedule makespan. Turek et al. [13] derive a 2–approximation algorithm for this problem. In fact, the model considered in [13] is just a special case of the problem considered in this paper. Interpreting the parallel processors as a generic 'resource' that must be allocated to jobs, the problem of [13] corresponds to the problem considered in this paper,

letting n jobs, with resource dependent processing times, be processed on $m = n$ *identical* parallel machines (instead of unrelated parallel machines). Mounie et al. [11] consider yet another variant, in that the processor allocations must be contiguous (for that problem, [13] includes a 2.7–approximation). Moreover, in [11] it is not only assumed that the processing times p_{js} are non-increasing in s, but also the processing 'areas' $s \cdot p_{js}$ are assumed to be non-decreasing in s. For that problem, a $\sqrt{3}$–approximation is derived in [11].

When we restrict even further, and assume that the decision on the allocation of resources to jobs is fixed beforehand, we are back at (machine) scheduling under resource constraints as introduced by Blazewicz et al. [1]. More recently, such problems with dedicated machines have been discussed by Kellerer and Strusevich [8, 9]. We refer to these papers for various complexity results, and note that NP-hardness of the problem with dedicated machines and a binary resource was established in [8]. More precisely, they show weak NP-hardness for the case where the number of machines is fixed, and strong NP-hardness for an arbitrary number of machines [8].

From the more practical viewpoint, a resource-time tradeoff problem has also been addressed by Chen [2]. He considers parallel, identical machine scheduling problems. Just like in the paper by Shmoys and Tardos [12], job processing times can be reduced by utilizing a non-renewable, monetary resource. The objective considered in [2] is either the total weighted completion times of jobs, or the weighted number of tardy jobs, incremented by the total resource consumption of the schedule. The paper describes branch and bound algorithms, together with computational results.

Results and Methodology. We derive a constant-factor approximation algorithm for the problem at hand. Our approach is based upon an integer linear program that defines a relaxation of the problem, extending a formulation used by Grigoriev et al. [7]. The main idea is the utilization of an aggregate version of the resource constraints, yielding a formulation that does not require time-indexed variables. More precisely, we use a formulation that takes as input all possible processing times p_{ijs} of jobs. We then consider the linear programming relaxation of this integer program. In a first step, the solution of this LP relaxation is rounded into a (still fractional) solution for another linear program. We then show that this in fact defines an instance (and solution) of the linear programming relaxation used by Shmoys and Tardos [12] for the generalized assignment problem. In a second step, we thus apply their rounding procedure to obtain an approximate integral solution for the original integer programming relaxation. From this solution, we extract both the machine assignments and the resource allocations for the jobs. We then use Grahams list scheduling [4] to generate a feasible schedule. Using the LP lower bounds, we prove that this schedule is not more than a factor $(4 + 2\sqrt{2})$ away from the optimum. For the special case of dedicated machines, our approach simplifies, because the machine assignments are fixed beforehand, thus the second rounding step is not required. For that case, we prove a performance bound of $(3 + 2\sqrt{2})$.

For both problems, unrelated and dedicated machines, we furthermore provide an instance showing that the linear programming based analysis cannot yield anything better than a 2 approximation.

Concerning lower bounds on the approximability, note that the problem at hand is a generalization of the classical unrelated machine scheduling problem $R||C_{\max}$. Therefore it cannot be approximated better than a multiplicative factor of 3/2, unless P=NP [10]. Restricting to the special case of dedicated machines, strong NP-hardness follows from Kellerer and Strusevich [8], hence the problem cannot admit an FPTAS, unless P=NP. This is true even for the case of a binary resource, i.e., if $k = 1$. We furthermore show that, for general k, the problem cannot be approximated better than a multiplicative factor of 3/2, unless P=NP.

2 Problem Definition

Let $V = \{1, \ldots, n\}$ be a set of jobs. Jobs must be processed non-preemptively on a set of m unrelated machines, and the objective is to find a schedule that minimizes the makespan $C_{\max}$, that is, the time of the last job completion. During its processing, a job j may be assigned an amount $s \in \{0, 1, \ldots, k\}$ of an additional resource, for instance additional workers, that may speed up its processing. If s resources are allocated to a job j, and the job is processed on machine i, the processing time of that job is p_{ijs}. The only assumption on the processing times in dependence on the amount of allocated resources is monotonicity, that is, we assume that

$$p_{ij0} \geq p_{ij1} \geq \cdots \geq p_{ijk}$$

for every machine-job pair (i, j). The allocation of resources to jobs is restricted as follows. At any time, no more than the available k units of the resource may be allocated to the set of jobs in process. Moreover, the amount of resources assigned to any job must be the same along its processing. In other words, if $s \leq k$ resources are allocated to some job j, and x_j denotes its starting time on some machine i, only $k - s$ of the resources are available for other jobs between x_j and $x_j + p_{ijs}$.

We finally introduce an additional piece of notation. Since we do not assume that the functions p_{ijs}, in dependence on s, are *strictly* decreasing, the only information that is effectively required is the *breakpoints* of p_{ijs}, this is, indices s where $p_{ijs} < p_{ij,s-1}$. Hence, define the 'relevant' indices for job j on machine i as

$$S_{ij} = \{0\} \cup \{s \mid s \leq k,\ p_{ijs} < p_{ij,s-1}\} \subseteq \{0, \ldots, k\}\,.$$

Considering this index set suffices, since in any solution, if s resources are allocated to some job j on machine i, we may as well use $s' = \min\{r \mid r \leq s,\ p_{ijr} = p_{ijs}\} \in S_{ij}$ resources for that job, without violating feasibility.

3 IP Relaxation and LP-Based Rounding

Let x_{ijs} denote binary variables, indicating that an amount of s resources is used for processing job j on machine i. Then the following integer linear program, referred to as (IP), has a feasible solution if there is a feasible schedule of length C for the original scheduling problem.

$$\sum_{i=1}^{m} \sum_{s \in S_{ij}} x_{ijs} = 1 \ , \qquad \forall\, j \in V \tag{1}$$

$$\sum_{j \in V} \sum_{s \in S_{ij}} x_{ijs}\, p_{ijs} \le C \ , \qquad \forall\, i = 1, \ldots, m\,, \tag{2}$$

$$\sum_{j \in V} \sum_{i=1}^{m} \sum_{s \in S_{ij}} x_{ijs}\, s\, p_{ijs} \le k\, C \ , \tag{3}$$

$$x_{ijs} = 0 \ , \qquad \text{if } p_{ijs} > C, \tag{4}$$

$$x_{ijs} \in \{0, 1\} \ , \qquad \forall\, i, j, s.$$

Here, C represents the schedule makespan. Equalities (1) make sure that every job is assigned to one machine and uses a constant amount of resources during its processing. Inequalities (2) express the fact that the total processing on each machine is a lower bound on the makespan. Inequalities (3) represent the aggregated resource constraints: In any feasible schedule, the left-hand side of (3) is the total resource consumption of the schedule. Because no more than k resources may be consumed at any time, the total resource consumption cannot exceed $k\,C$. Finally, constraints (4) make sure that we do not use machine-resource pairs such that the job processing time exceeds the schedule makespan. These constraints are obviously redundant for (IP), but they will be used later in rounding a solution for the linear relaxation of (IP). Notice that this integer program may have a feasible solution for some integer value of C, although no feasible schedule with makespan C exists; see Example 1 further below.

LP Relaxation. The integer linear program (IP) with the 0/1-constraints on x relaxed to

$$x_{ijs} \ge 0\,, \qquad j \in V\,,\ s \in S_{ij}\,,\ i = 1, \ldots, m$$

also has a solution of value at most C if there is a feasible schedule for the original scheduling problem with makespan C. We refer to this relaxation as (LP), and note that it can be solved in polynomial time, because it has a polynomial number of variables and constraints. Since we assume integrality of data, we are actually only interested in integral values C. Therefore, by using binary search, we can find in polynomial time the smallest integral value C^{LP} such that (LP) has a feasible solution x^{LP}. Then C^{LP} is a lower bound on the makespan of any feasible schedule.

Rounding the LP Solution. Given a pair $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ we next define an integer solution x^* from x^{LP} by the following, 2-phase rounding procedure. In the first rounding phase, we transform a fractional solution x^{LP} to another fractional solution $\bar{x}$, in such a way that for every machine-job pair (i, j) there is exactly one index s (amount of resource) such that $\bar{x}_{ijs}$ is nonzero. Intuitively, we decide for every machine-job pair on the amount of resources it may consume. By doing this, we effectively get rid of the index s. This new fractional solution in fact defines a fractional solution for an LP relaxation for the generalized assignment problem discussed by Shmoys and Tardos [12]. Therefore, we will be able to use their rounding procedure as our second rounding phase, and thus we eventually obtain an integral solution x^* from x^{LP}.

First, let us choose an arbitrary ε such that $0 \leq \varepsilon \leq 1$. Then, for every machine i and job j individually, define

$$\tilde{y}_{ij} = \sum_{s \in S_{ij}} x^{\mathrm{LP}}_{ijs} \tag{5}$$

as the total fractional value allocated by the LP solution x^{LP} to the machine-job pair (i, j). Then let index $t^{ij} \in S_{ij}$ be chosen minimal with the property that

$$\sum_{s \in S_{ij}, s \leq t^{ij}} x^{\mathrm{LP}}_{ijs} \geq (1 - \varepsilon)\, \tilde{y}_{ij}\,. \tag{6}$$

Then, for every machine i and job j define index $s^{ij} \geq t^{ij}$ as the minimizer of $s \cdot p_{ijs}$, for $s \geq t^{ij}$,

$$s^{ij} = \arg\min_{s \geq t^{ij}} \; s \cdot p_{ijs}\,. \tag{7}$$

By definition, it follows that $s^{ij} \in S_{ij}$. We now consider a fractional solution $\bar{x}$ defined by

$$\bar{x}_{ijs} = \begin{cases} \tilde{y}_{ij} & s = s^{ij}\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{8}$$

By definition, this solution fulfills (1). Moreover, we claim that it is an approximate solution for inequalities (2) and (3) in the following sense.

Lemma 1. *Let $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ be an optimal fractional solution for the linear programming relaxation* (LP)*, and let $\bar{x} = (\bar{x}_{ijs})$ be the fractional solution obtained by the above described rounding procedure. Then*

$$\sum_{j \in V} \sum_{s \in S_{ij}} \bar{x}_{ijs}\, p_{ijs} \leq \frac{1}{1 - \varepsilon}\, C^{\mathrm{LP}}\,, \qquad i = 1, \ldots, m\,, \tag{9}$$

$$\sum_{j \in V} \sum_{i=1}^{m} \sum_{s \in S_{ij}} \bar{x}_{ijs}\, s\, p_{ijs} \leq \frac{k}{\varepsilon}\, C^{\mathrm{LP}}\,. \tag{10}$$

Proof. The proof of both claims is based on proving the statement for every machine-job pair. Validity of (9) can be seen as follows. We know that

$$(1-\varepsilon)\,\bar{x}_{ijs^{ij}} = (1-\varepsilon)\,\tilde{y}_{ij} \leq \sum_{s\in S_{ij}, s\leq t^{ij}} x^{\mathrm{LP}}_{ijs}$$

by definition of t^{ij} in (6). By the fact that $p_{ijt^{ij}} \leq p_{ijs}$ for all $s \leq t^{ij}$, we therefore have

$$(1-\varepsilon)\,\bar{x}_{ijs^{ij}} p_{ijt^{ij}} \leq \sum_{s\in S_{ij}, s\leq t^{ij}} x^{\mathrm{LP}}_{ijs}\, p_{ijs} \leq \sum_{s\in S_{ij}} x^{\mathrm{LP}}_{ijs}\, p_{ijs}$$

for every machine i and job $j \in V$. Again due to monotonicity, $p_{ijs^{ij}} \leq p_{ijt^{ij}}$ for all $j \in V$ and $i = 1, \ldots, m$, and we obtain

$$\begin{aligned}\sum_{s\in S_{ij}} \bar{x}_{ijs}\, p_{ijs} \;&=\; \bar{x}_{ijs^{ij}} p_{ijs^{ij}} \;\leq\; \bar{x}_{ijs^{ij}} p_{ijt^{ij}} \\ &\leq\; \frac{1}{1-\varepsilon} \sum_{s\in S_{ij}} x^{\mathrm{LP}}_{ijs}\, p_{ijs}\end{aligned}$$

for all jobs $j \in V$ and machines $i = 1, \ldots, m$. Summing over $j \in V$, and using (2), inequalities (9) follow for any machine i.

To see (10), first observe that $\varepsilon\,\bar{x}_{ijs^{ij}} = \varepsilon\,\tilde{y}_{ij} \leq \sum_{s\in S_{ij}, s\geq t^{ij}} x^{\mathrm{LP}}_{ijs}$ by definition of t^{ij}, since t^{ij} is the *minimal* index with property (6). Therefore,

$$\varepsilon\,\bar{x}_{ijs^{ij}} s^{ij} p_{ijs^{ij}} \leq \sum_{s\in S_{ij}, s\geq t^{ij}} x^{\mathrm{LP}}_{ijs}\, s\, p_{ijs} \leq \sum_{s\in S_{ij}} x^{\mathrm{LP}}_{ijs}\, s\, p_{ijs}$$

for every machine i and job $j \in V$, where the first inequality follows because s^{ij} was chosen among all $s \geq t^{ij}$ such as to minimize $s\, p_{ijs}$. Hence, we obtain

$$\sum_{s\in S_{ij}} \bar{x}_{ijs}\, s\, p_{ijs} = \bar{x}_{ijs^{ij}} s^{ij}\, p_{ijs^{ij}} \leq \frac{1}{\varepsilon} \sum_{s\in S_{ij}} x^{\mathrm{LP}}_{ijs}\, s\, p_{ijs}\,,$$

for all jobs $j \in V$ and machines $i = 1, \ldots, m$. Summing over $j \in V$ and all machines $i = 1, \ldots, m$, and using (3), eventually yields (10). □

Next, we want to use the rounding procedure by Shmoys and Tardos in order to end up with an integer solution.

Lemma 2 (Shmoys & Tardos [12–Theorem 2.1]). *Given a feasible fractional solution $\tilde{y} = (\tilde{y}_{ij})$ to the linear program*

$$\sum_{i=1}^{m} y_{ij} = 1 \ , \qquad \forall\ j \in V \tag{11}$$

$$\sum_{j \in V} y_{ij}\tau_{ij} \le T \ , \qquad \forall\ i = 1, \ldots, m\,, \tag{12}$$

$$\sum_{j \in V} \sum_{i=1}^{m} y_{ij}\lambda_{ij} \le \Lambda \ , \tag{13}$$

$$y_{ij} \ge 0 \ , \qquad \forall\ i, j. \tag{14}$$

with nonnegative parameters $T, \Lambda, \tau = (\tau_{ij})$, and $\lambda = (\lambda_{ij})$, there is a polynomial time algorithm which computes an integral solution $\bar{y}$ to (11), (13), (14), *and*

$$\sum_{j \in V} \bar{y}_{ij}\tau_{ij} \le T + \tau_{max} \ , \qquad \forall\ i = 1, \ldots, m\,, \tag{15}$$

where $\tau_{\max} = \max_{i,j}\{\tau_{ij} \mid \tilde{y}_{ij} > 0\}$. □

The fractional solution $\tilde{y}$ defined in (5), however, is nothing but a feasible fractional solution for linear program (11)–(14), namely with parameters $T = 1/(1-\varepsilon)\, C^{\mathrm{LP}}$, $\Lambda = k/\varepsilon\, C^{\mathrm{LP}}$, $\tau_{ij} = p_{ijs^{ij}}$, and $\lambda_{ij} = s^{ij} p_{ijs^{ij}}$ for all job-machine pairs (i, j). Therefore, combining Lemma 1, the above result of Shmoys and Tardos, and the fact that

$$\tau_{\max} = \max_{i,j} p_{ijs^{ij}} \le \max_{i,j,s}\{p_{ijs} \mid x^{\mathrm{LP}}_{ijs} > 0\} \le C^{\mathrm{LP}} \tag{16}$$

by constraints (4), we can show the following.

Lemma 3. *Let $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ be an optimal fractional solution for the linear programming relaxation* (LP)*, then we can find a feasible solution $x^* = (x^*_{ijs})$ for the following integer linear program in polynomial time.*

$$\sum_{i=1}^{m} \sum_{s \in S_{ij}} x_{ijs} = 1 \ , \qquad \forall\ j \in V, \tag{17}$$

$$\sum_{j \in V} \sum_{s \in S_{ij}} x_{ijs}\, p_{ijs} \le \left(1 + \frac{1}{1-\varepsilon}\right) C^{\mathrm{LP}} \ , \qquad \forall\ i\,, \tag{18}$$

$$\sum_{j \in V} \sum_{i=1}^{m} \sum_{s \in S_{ij}} x_{ijs}\, s\, p_{ijs} \le \frac{k}{\varepsilon}\, C^{\mathrm{LP}} \ , \tag{19}$$

$$x_{ijs} \in \{0, 1\} \ , \qquad \forall\ i, j, s\,. \tag{20}$$

Proof. We briefly summarize the previously described steps. Using the fractional solution $x^{\mathrm{LP}} = (x^{\mathrm{LP}}_{ijs})$, define $\tilde{y} = (\tilde{y}_{ij})$ as in (5), and apply the rounding defined

by (8). This yields a fractional solution $\bar{x} = (\bar{x}_{ijs})$ that is nonzero only for one resource index $s = s^{ij}$, for any pair of i and j, as defined in (7). Interpreting $\tilde{y}$ as fractional solution for the generalized assignment problem (11)–(14), use Lemma 2 to round it to an integral solution $\bar{y} = (\bar{y}_{ij})$. Now define the integral solution x^* by

$$x^*_{ijs} = \begin{cases} \bar{y}_{ij} & s = s^{ij}\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

With the help of Lemmas 1 and 2, and utilizing (16), it is now straightforward to verify that x^* fulfills (17)–(20). □

4 LP Based Greedy Algorithm

Our approach to obtain a constant factor approximation for the scheduling problem is now the following. We first use the rounded 0/1-solution from the previous section in order to decide both, on the amount of resources allocated to every individual job j, and on the machine where this job must be executed. More precisely, job j must be processed on machine i and use s additional resources iff $x^*_{ijs} = 1$, where x^* is the feasible integral solution of (17)–(20) obtained after the 2-phase rounding. Then the jobs are scheduled according to the greedy list scheduling algorithm of Graham [4], in arbitrary order.

> **Algorithm** LP-Greedy: With the resource allocations and machine assignments as determined by the LP based rounding, do until all jobs are scheduled: Starting at time 0, iterate over completion times of jobs, and schedule as many jobs as allowed, obeying the machine assignments and the resource constraints.

Theorem 1. *Algorithm* LP-Greedy *is a* $(4 + 2\sqrt{2})$*–approximation algorithm for unrelated parallel machine scheduling with resource dependent processing times.*

The fact that the algorithm requires only polynomial time follows directly from the fact that both, solving and rounding the LP relaxation, as well as the list scheduling, can be implemented in polynomial time.

To verify the performance bound, we first need some additional notation. Consider some schedule $\mathcal{S}$ produced by algorithm LP-Greedy, and denote by C^{LPG} the corresponding makespan. Denote by C^{OPT} the makespan of an optimal solution. For schedule $\mathcal{S}$, let $t(\beta)$ denote the earliest point in time after which only *big jobs* are processed, big jobs being defined as jobs that have a resource consumption larger than $k/2$. Moreover, let $\beta = C^{\mathrm{LPG}} - t(\beta)$ be the length of the period in which only big jobs are processed (note that possibly $\beta = 0$).

Next, we fix a machine, say machine i, on which some job completes at time $t(\beta)$ which is not a big job. Due to the definition of $t(\beta)$, such a machine

must exist, because otherwise all machines were idle right before $t(\beta)$, contradicting the definition of the greedy algorithm. Note that, between time 0 and $t(\beta)$, periods may exist where machine i is idle. Denote by α the total length of busy periods on machine i between 0 and $t(\beta)$, and by γ the total length of idle periods on machine i between 0 and $t(\beta)$. We then have that

$$C^{\mathrm{LPG}} = \alpha + \beta + \gamma\,. \tag{21}$$

Due to (18), we get that for machine i

$$\alpha \leq \sum_{j \in V} \sum_{s \in S_{ij}} x^*_{ijs}\, p_{ijs} \leq \left(1 + \frac{1}{1-\varepsilon}\right) C^{\mathrm{LP}}\ . \tag{22}$$

The next step is an upper bound on $\beta + \gamma$, the length of the final period where only big jobs are processed, together with the length of idle periods on machine i.

Lemma 4. *We have that*

$$\beta + \gamma \leq \frac{2}{\varepsilon}\, C^{\mathrm{LP}}\,.$$

Proof. First, observe that the total resource consumption of schedule $\mathcal{S}$ is at least $\beta\,\frac{k}{2} + \gamma\,\frac{k}{2}$. This because, on the one hand, all jobs after $t(\beta)$ are big jobs and require at least $k/2$ resources, by definition of $t(\beta)$. On the other hand, during all idle periods on machine i between 0 and $t(\beta)$, at least $k/2$ of the resources must be in use as well. Assuming the contrary, there was an idle period on machine i with at least $k/2$ free resources. But after that idle period, due to the selection of $t(\beta)$ and machine i, some job is processed on machine i which is not a big job. This job could have been processed earlier during the idle period, contradicting the definition of the greedy algorithm. Next, recall that $(k/\varepsilon)\, C^{\mathrm{LP}}$ is an upper bound on the total resource consumption of the jobs, due to (19). Hence, we obtain

$$\frac{k}{\varepsilon}\, C^{\mathrm{LP}} \geq \beta\,\frac{k}{2} + \gamma\,\frac{k}{2}\,.$$

Dividing by $2/k$ yields the claimed bound on $\beta + \gamma$. □

Now we are ready to prove the performance bound of Theorem 1.

Proof (of Theorem 1). First, use (21) together with (22) and Lemma 4 to obtain

$$C^{\mathrm{LPG}}\ \leq \left(1 + \frac{1}{1-\varepsilon}\right) C^{\mathrm{LP}} + \frac{2}{\varepsilon}\, C^{\mathrm{LP}}\ \leq \left(1 + \frac{1}{1-\varepsilon} + \frac{2}{\varepsilon}\right) C^{\mathrm{OPT}}\,.$$

Solving for the best possible value for ε gives $\varepsilon = 2 - \sqrt{2} \approx 0.5858$, which yields the claimed performance bound of $4 + 2\sqrt{2}$. □

5 Dedicated Machines

As a special case of the unrelated machine scheduling model considered so far, let us assume that the jobs are assigned to machines *beforehand*. That is, the set of jobs V is partitioned into m subsets $V_1, \ldots, V_m$ a priori, V_i being the jobs that must be processed on machine i, and p_{js}, $s = 0, \ldots, k$, denotes the resource dependent processing time of job j.

By letting all but one machine assignment result in very large processing times, this is obviously a special case of the unrelated machine scheduling model. Hence, our above analysis also yields a $(4+2\sqrt{2})$–approximation for this model. However, noting that the machine index i can be eliminated from the linear program, we can use the following LP relaxation instead.

$$\text{min.} \quad C \tag{23}$$

$$\text{s. t.} \quad \sum_{s \in S_j} x_{js} = 1 \ , \qquad \forall\, j \in V\,, \tag{24}$$

$$\sum_{j \in V_i} \sum_{s \in S_j} x_{js}\, p_{js} \le C \ , \qquad \forall\, i = 1, \ldots, m\,, \tag{25}$$

$$\sum_{j \in V} \sum_{s \in S_j} x_{js}\, s\, p_{js} \le k\, C \ , \tag{26}$$

$$x_{js} \ge 0 \ , \qquad \forall j, s\,. \tag{27}$$

Here, according to our previous notation, S_j are the breakpoints of the function p_{js}, hence

$$S_j = \{0\} \cup \{s \mid s \le k,\ p_{js} < p_{j,s-1}\} \subseteq \{0, \ldots, k\}\,.$$

This way, the accordingly adapted first phase rounding of (8) already yields an integral solution. More precisely, given a fractional solution $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ for the above LP relaxation, we choose index t^j minimal with the property that $\sum_{s \in S_j, s \le t^j} x^{\mathrm{LP}}_{js} \ge 1 - \varepsilon$ and define index $s^j = \arg\min_{s \ge t^j} s \cdot p_{js}$. Again, it follows that $s^j \in S_j$. Then the solution $\bar{x}$, defined by $\bar{x}_{js} = 1$ if $s = s^j$ and $\bar{x}_{js} = 0$ otherwise, is already integral. Hence, Shmoys and Tardos' rounding is not required, and instead of using the bounds (18) and (19), we now have an integral solution $\bar{x} = (\bar{x}_{js})$ which fulfills the constraints

$$\sum_{j \in V_i} \sum_{s \in S_j} \bar{x}_{js}\, p_{js} \le \frac{1}{1-\varepsilon}\, C \ , \qquad \forall\, i = 1, \ldots, m\,,$$

$$\sum_{j \in V} \sum_{s \in S_j} \bar{x}_{js}\, s\, p_{js} \le \frac{k}{\varepsilon}\, C \ .$$

Validity of these bounds is proved along the same lines as Lemma 1. This eventually yields an improved performance bound for the dedicated machine model.

Theorem 2. *Algorithm* LP-GREEDY *is a* $(3+2\sqrt{2})$*–approximation algorithm for dedicated parallel machine scheduling with resource dependent processing times.*

Lower Bound for the (Integer) Linear Program. We next give an instance to show that the integer linear program we use can be a factor 2 away from the optimal solution. Hence, our LP-based analysis cannot yield anything better than a 2-approximation, for both versions, the dedicated and the unrelated machine case.

Example 1. *Consider the problem with* $m = 2$ *dedicated machines and* k *units of the additional resource, where* k *is odd. There are* 2 *jobs, each to be processed on its own machine, with resource-dependent processing times*

$$p_{js} = \begin{cases} 2k+1 & \text{if } s < \frac{k}{2} \\ k & \text{if } s > \frac{k}{2} \end{cases}$$

for both jobs j. □

We have the following, feasible integer solution for the LP-relaxation (23)–(27): $x_{js} = 1$ if $s = \lceil k/2 \rceil$, and $x_{js} = 0$ otherwise, for both jobs j. This setting of variables would yield $C \geq k$ by (25), and $kC \geq 2k\lceil k/2 \rceil = k(k+1)$ by (26). Therefore, with $C = (k+1)$, there exists a feasible, even integral, solution for (23)–(27). A fortiori, we know that for the linear programming relaxation (LP), $C^{\mathrm{LP}} \leq k+1$. But in the optimal solution, $C^{\mathrm{OPT}} = 2k$. Hence, the gap between C^{LP} and C^{OPT} can be as large as $2-\varepsilon$, for any $\varepsilon > 0$. The bad quality of the LP lower bound is obviously a consequence of the fact that we only use an aggregate formulation of the resource constraints in (26) (or (3), respectively), whereas any schedule has to respect the resource constraint at any time.

6 Lower Bounds on Approximation

The problem with unrelated machines cannot be approximated within a factor smaller than 3/2 as a generalization of the classical unrelated machine scheduling problem [10], as mentioned earlier. We next show that the same inapproximability result holds for the problem with dedicated machines.

Theorem 3. *There is no polynomial time approximation algorithm for dedicated parallel machine scheduling with resource dependent processing times that has a performance guarantee less than* 3/2*, unless* P=NP.

Proof. The proof relies on a gap-reduction from PARTITION [3]: Given n integers a_j, with $\sum_{j=1}^{n} a_j = 2k$, it is NP-complete to decide if there exists a subset $W \subseteq \{1,\ldots,n\}$ with $\sum_{j\in W} a_j = k$. Let us define an instance of the dedicated machine scheduling problem as follows. Each a_j gives rise to one job j with an individual machine. Hence, we have n jobs and $m = n$ dedicated machines. There

are k units available of the additional resource. Any job j has a processing time defined by

$$p_{js} = \begin{cases} 3 & \text{if } s < a_j \\ 1 & \text{if } s \geq a_j \,. \end{cases}$$

Hence, the a_j's are the only breakpoints in the functions p_{js}, and the index set $S_j = \{0, a_j\}$ for all jobs j. In other words, the functions p_{js} can be encoded in $\mathrm{O}(\log a_j)$ for all jobs j, and the transformation is indeed polynomial. We claim that there exists a feasible schedule with makespan $C_{\max} < 3$ if and only if there exists a solution for the PARTITION problem. Otherwise, the makespan is at least 3. To this end, observe that in any solution with makespan $C_{\max} < 3$, we may assume that each job j consumes exactly a_j units of the resource: If it was less than a_j for some jobs j, the makespan would be at least 3; if it was more than a_j for some job j, letting the resource allocation equal a_j does not violate feasibility, while maintaining the same processing time. Now, if and only if there is a solution, say W, for the PARTITION problem, there exists a resource feasible schedule with makespan 2, namely where jobs $j \in W$ start at time 0, and all jobs $j \notin W$ start at time 1. □

Finally, it is not difficult to see that the above proof yields the same inapproximability result for the problem with dedicated machines, even if the resource consumption of jobs is fixed beforehand.

Corollary 1. *There is no polynomial time approximation algorithm for resource constrained dedicated machine scheduling that has a performance guarantee less than* 3/2*, unless* P=NP*.*

References

1. J. BLAZEWICZ, J. K. LENSTRA AND A. H. G. RINNOOY KAN, Scheduling subject to resource constraints: Classification and complexity, *Discrete Applied Mathematics*, **5** (1983), pp. 11–24.
2. Z.-L. CHEN, Simultaneous Job Scheduling and Resource Allocation on Parallel Machines, *Annals of Operations Research*, **129** (2004), pp. 135-153.
3. M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completenes*, W. H. Freeman, New York, 1979.
4. R. L. GRAHAM, Bounds for certain multiprocessing anomalies, *Bell System Technical Journal*, **45** (1966), pp. 1563–1581. See also [5].
5. R. L. GRAHAM, Bounds on multiprocessing timing anomalies, SIAM *Journal on Applied Mathematics*, **17** (1969), pp. 416–429.
6. R. L. GRAHAM, E. L. LAWLER, J. K. LENSTRA, AND A. H. G. RINNOOY KAN, Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics*, **5** (1979), pp. 287–326.
7. A. GRIGORIEV, H. KELLERER AND V. A. STRUSEVICH, Scheduling parallel dedicated machines with the speeding-up resource, manuscript (2003). Extended abstract in: *Proceedings of the 6th Workshop on Models and Algorithms for Planning and Scheduling Problems*, Aussois, France, 2003, pp. 131–132.

8. H. Kellerer and V. A. Strusevich, Scheduling parallel dedicated machines under a single non-shared resource, *European Journal of Operational Research*, **147** (2003), pp. 345–364.
9. H. Kellerer and V. A. Strusevich, Scheduling problems for parallel dedicated machines under multiple resource constraints, *Discrete Applied Mathematics*, **133** (2004), pp. 45–68.
10. J. K. Lenstra, D. B. Shmoys and E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming*, Series A, **46** (1990), pp. 259–271.
11. G. Mounie, C. Rapine, and D. Trystram, Efficient Approximation Algorithms for Scheduling Malleable Tasks, *Proceedings of the 11th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1999, pp. 23–32.
12. D. B. Shmoys and E. Tardos, An approximation algorithm for the generalized assignment problem, *Mathematical Programming*, Series A, **62** (1993), pp. 461–474.
13. J. Turek, J. L. Wolf, and P. S. Yu, Approximate Algorithms for Scheduling Parallelizable Tasks, *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1992, pp. 323–332.

LP-Based Online Scheduling: From Single to Parallel Machines

José R. Correa[1] and Michael R. Wagner[2]

[1] School of Business, Universidad Adolfo Ibáñez, Santiago, Chile
correa@uai.cl
[2] Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139
mikew@mit.edu

Abstract. We study classic machine sequencing problems in an online setting. Specifically, we look at deterministic and randomized algorithms for the problem of scheduling jobs with release dates on identical parallel machines, to minimize the sum of weighted completion times: Both preemptive and non-preemptive versions are analyzed. Using linear programming techniques, borrowed from the single machine case, we are able to design a 2.62-competitive deterministic algorithm for the non-preemptive version of the problem, improving upon the 3.28-competitive algorithm of Megow and Schulz [13]. Additionally, we obtain randomized algorithms for both versions of the problem with competitive ratio strictly smaller than 2 for any number of machines (but approaching two as the number of machines grows). Our algorithms naturally extend several approaches for single and parallel machine scheduling. Moreover, in contrast to previous algorithms achieving similar bounds, obtained by Schulz and Skutella [15], ours is a list-scheduling algorithm and has one less randomization step.

1 Introduction

We study online versions of classic parallel machine scheduling problems. Given a set of jobs $N = \{1, \ldots, n\}$, where each job j has a processing time $p_j > 0$, a weight $w_j > 0$ and a release date $r_j \geq 0$, we want to process these jobs on m identical machines. We consider both non-preemptive and preemptive versions; in the latter case, a job being processed may be interrupted and resumed later, possibly on a different machine. Letting C_j be the completion time of job j in a given schedule, we are interested in minimizing the weighted sum of completion times: $\sum_{j \in N} w_j C_j$. As we only consider online problems, jobs are not known until their respective release date. In scheduling notation [10], we consider online versions of $P|r_j| \sum_j w_j C_j$ and $P|r_j, pmtn| \sum_j w_j C_j$.

In online optimization we are dealing with limitations on information, contrasting with the limitations on computational power in classic approximation algorithm design. The standard measure of quality of online algorithms is the

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 196–209, 2005.

so-called competitive ratio. Similarly to the approximation guarantee of an algorithm, this is defined to be the worst case ratio, over all instances, of the cost output by the online algorithm to the optimal offline cost. We shall also say that an online algorithm is c-competitive if, for any instance, the cost output by the online algorithm is at most c times the optimal offline cost. We may alternatively define the competitive ratio of an online algorithm as the infimum of all values c, where the algorithm is c-competitive. In some situations randomization is a powerful tool to obtain algorithms with better performance ratios. The competitive ratio of a randomized online algorithm is defined as above replacing "the cost output by the online algorithm" by the *expected* cost output by the online algorithm.

1.1 Previous Work

There has been an enormous amount of work on parallel machine scheduling. As we do not intend to do a complete review of results in the area, let us only mention some of the most relevant literature on online scheduling problems directly related to the matter of this paper.

To the best of our knowledge, the first deterministic online algorithm for $P|r_j|\sum_j w_jC_j$ was given by Hall, Schulz, Shmoys and Wein [11]. They design a $(4+\varepsilon)$-competitive algorithm. Prior to this paper, the best-known deterministic algorithms for both $P|r_j|\sum_j w_jC_j$ and $P|r_j, pmtn|\sum_j w_jC_j$ were recently given by Megow and Schulz [13] and are 3.28 and 2-competitive, respectively. They also show that the former algorithm has a competitive ratio between 2.78 and 3.28 while the latter analysis is tight.

Considering randomized algorithms, a $(2.89+\varepsilon)$-competitive algorithm for $P|r_j|\sum w_jC_j$, was obtained by Chakrabarti, Phillips, Schulz, Shmoys, Stein and Wein [3]. Schulz and Skutella [15] give randomized strategies that are 2-competitive for both $P|r_j, pmtn|\sum_j w_jC_j$ and $P|r_j|\sum_j w_jC_j$. Related results have been obtained in [4, 14] when the objective is to minimize the average completion time of the schedule. In a more restricted setting, Chou, Queyranne and Simchi-Levi [5] consider the online $P|r_j|\sum_j w_jC_j$ with lower and upper bounds on jobs' weights and processing times; the authors prove that the online *weighted shortest processing time* heuristic is asymptotically optimal. They even extend this to the problem $Q|r_j|\sum_j w_jC_j$.

We also mention some single machine scheduling results, as our work essentially extends these analyses to the parallel machine case. Using the idea of α-points and mean-busy-time relaxations, Goemans, Queyranne, Schulz, Skutella and Wein [9] designed a deterministic 2.4143-competitive and a randomized 1.6853-competitive algorithm for the online $1|r_j|\sum_j w_jC_j$. A similar approach was taken by Schulz and Skutella [16] to give a randomized $\frac{4}{3}$-competitive algorithm for $1|r_j, pmtn|\sum_j w_jC_j$, which improved upon a 1.56-competitive algorithm by Sitters [18]. On the other hand, Anderson and Potts [2] provide a best possible deterministic online algorithm for $1|r_j|\sum_j w_jC_j$ which has a competitive ratio of 2.

Let us now discuss some lower bounds on the competitive ratios for certain problems. Hoogeveen and Vestjens [12] showed that there is no deterministic algorithm with competitive ratio strictly better than 2 for $1|r_j|\sum_j w_jC_j$. On the other hand Stougie and Vestjens [19] showed that $\frac{e}{e-1}$ is a lower bound on the competitive ratio of online randomized algorithms for the same problem. In the parallel machine case, Vestjens [20] proved that any deterministic algorithm for $P|r_j|\sum_j w_jC_j$ (resp. $P|r_j, pmtn|\sum_j w_jC_j$) has a competitive ratio of at least 1.309 (resp. $\frac{22}{21}$). Seiden [17] proved that any randomized algorithm for $P|r_j|\sum_j w_jC_j$ has a competitive ratio of at least 1.157. To the best of our knowledge, there are no specific lower bounds for randomized algorithms for $P|r_j, pmtn|\sum_j w_jC_j$.

Finally, we remark that all the problems considered in this paper admit polynomial time approximation schemes (offline) [1].

1.2 Our Results

Our main results are the following:

- A 2.618-competitive online algorithm for $P|r_j|\sum_j w_jC_j$, which improves upon the 3.28-competitive algorithm of Megow and Schulz [13].
- A randomized ϱ_m-competitive online algorithm for $P|r_j|\sum_j w_jC_j$, where $\varrho_m < 2$ for all $m \geq 1$. Here, m denotes the number of machines and ϱ_m is obtained implicitly. Our result improves upon the 2-competitive randomized algorithms by Schulz and Skutella [15]. In contrast to their work our algorithm has the desirable property of being a list-scheduling algorithm and uses only one step of randomization. The algorithm we present can be seen as the parallel machine extension of Goemans et al.'s [9] algorithm for a single machine. Indeed, the competitive ratio that it achieves is 1.6853 for $m = 1$ (as in Goemans et al.); for $m = 2, 3$ and 4 it is 1.8382, 1.8915 and 1.9184, respectively.
- A randomized ρ_m-competitive online algorithm for $P|r_j, pmtn|\sum_j w_jC_j$, where $\rho_m < 2$ for all $m \geq 1$. Here, m denotes the number of machines and ρ_m is obtained implicitly. The reader may wish to compare our result with the current best algorithm to date: the deterministic algorithm by Megow and Schulz [13], which has a competitive ratio of 2 (and not better than 2) for any number of machines. Additionally, our algorithm can be simultaneously seen as an extension of Megow and Schulz's result and of Schulz and Skutella's [16] single machine algorithm. Indeed for a single machine, the competitive ratio of our algorithm is 4/3, as in [16]; for two, three and four machines it is 1.3867, 1.603 and 1.7173, respectively. In general we have that $\rho_m < 2 - 1/m$, for $m > 1$.

2 Preliminaries

According to Phillips et al. [14] the α-point $t_j(\alpha)$, $0 < \alpha \leq 1$, of job j in a given schedule is defined as the first time an α-fraction of job j has been completed

(i.e., the first time αp_j has been processed). The general idea of our subsequent algorithms is to schedule jobs on the m machines by list-scheduling the jobs in the order of their α-points on a virtual machine, which is "m-times faster". Additionally, these algorithms may use job-dependent α's to guide the schedule; in this latter case, we shall denote job j's alpha value as α_j. The concept of a single fast virtual machine was apparently first considered by Eastman, Even and Isaacs [6]. Recently, Chekuri et al. [4] considered a "preemptive one-machine relaxation" where jobs are list-scheduled on parallel machines in order of their completion times on a single virtual machine.

Another important ingredient in what follows is related to mean-busy-time relaxations of $1|r_j|\sum w_jC_j$. The *mean busy time* M_j of job j is defined as the average point in time at which job j is being processed [8, 9]. Alternatively it can be computed as $M_j = \int_0^1 t_j(\alpha)d\alpha$. Let $p(S) = \sum_{j\in S} p_j$, $w(S) = \sum_{j\in S} w_j$ and $r_{\min}(S) = \min_{j\in S}\{r_j\}$. Following Goemans et al. [9] for a scheduling instance $I = \{(p_i, r_i, w_i),\ i \in N\}$ we define $Z_R(I)$ to be the value of mean busy time relaxation for $1|r_j, pmtn|\sum_j w_jC_j$; i.e.:

$$Z_R(I) \stackrel{\Delta}{=} \min \sum_{j\in N} w_jM_j$$

$$\text{subject to} \quad \sum_{j\in S} p_jM_j \geq p(S)\left(r_{\min}(S) + \frac{1}{2}p(S)\right),\ S \subseteq N.$$

It was shown in [8] that $Z_R(I)$ can be obtained online by scheduling, at any point in time, the available job j with the highest ratio w_j/p_j. This schedule is called the *LP schedule*.

Now, for an instance $I = \{(p_i, r_i, w_i),\ i \in N\}$ of $P|r_j, pmtn|\sum w_jC_j$ with m parallel machines, let $Z^m(I)$ be the value of the optimal schedule. Consider the instance $I_m = \{(\frac{p_i}{m}, r_i, w_i),\ i \in N\}$ and let $Z_R^m(I) = Z_R(I_m)$ i.e., the value of the mean-busy-time relaxation on I_m (note that this is equivalent to the value of the mean-busy-time relaxation on instance I in a machine that is m times faster). Thus, $Z_R^m(I)$ can be evaluated as:

$$Z_R^m(I) = \min \sum_{j\in N} w_jM_j$$

$$\text{subject to} \quad \sum_{j\in S} p_jM_j \geq p(S)\left(r_{\min}(S) + \frac{1}{2m}p(S)\right),\ S \subseteq N.$$

The following lemma provides a simple, yet powerful, lower bound for the problem $P|r_j, pmtn|\sum w_jC_j$. It is a particular case of a bound obtained by Chou et al. [5] in a more general framework. It was also obtained by Schulz and Skutella [15], expressed in terms of an equivalent time-indexed relaxation.

Lemma 1 ([5],[15]). *For any scheduling instance I,*

$$Z_R^m(I) + \frac{1}{2}\sum_{j\in N} w_jp_j \leq Z^m(I).$$

To finish this section let us review the concept of canonical decomposition [7] and a useful formula to rewrite $\sum w_j M_j$ [8] (see also [9]). For a set of jobs S, consider a single machine schedule that processes jobs in S as early as possible. This induces a partition of jobs in S into $S_1, \ldots, S_k$ such that the machine is busy exactly in the disjoint intervals $[r_{\min}(S_l), r_{\min}(S_l) + p(S_l)]$, for $l = 1, \ldots, k$. This partition is the *canonical decomposition* of S. Also, a set is canonical if it equals its canonical decomposition. Assume that $w_1/p_1 \geq \cdots \geq w_n/p_n \geq w_{n+1}/p_{n+1} = 0$ and let $[i] = \{1, \ldots, i\}$. Consider $S_1^i, \ldots, S_{k(i)}^i$, the canonical decomposition of $[i]$; then for any vector $M = (M_1, \ldots, M_n)$:

$$\sum_{j \in N} w_j M_j = \sum_{i=1}^{n} \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}} \right) \sum_{j \in [i]} p_j M_j = \sum_{i=1}^{n} \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}} \right) \sum_{l=1}^{k(i)} \sum_{j \in S_l^i} p_j M_j. \tag{1}$$

3 A Deterministic Online Algorithm for $P|r_j| \sum w_j C_j$

Consider the following online algorithm *Non-preemptive α Scheduling* (NAS), where each job j is assigned a deterministic value of α_j:

Algorithm NAS:

INPUT: A scheduling instance $I = \{(p_i, r_i, w_i),\ i \in N\}$ which is revealed online, and a vector $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_n\}$.

(1) Construct the preemptive LP-schedule on a single virtual machine m-times faster ($I \mapsto I_m$).

(2) At job j's α_j-point $t_j(\alpha_j)$ in the virtual machine, it enters into a FIFO queue for the m machines (job j is then scheduled the first time a machine is available after all preceding jobs in the queue have started).

From now on, whenever we refer to the LP-schedule of instance I, we mean the LP-schedule in a machine that is m times faster (or the LP-schedule of I_m). Consider job j and let $\eta_k(\alpha_j)$ denote the fraction of job k that has been completed in the LP-schedule by time $t_j(\alpha_j)$. Letting $C_j^{\boldsymbol{\alpha}}$ denote the completion time of job j in algorithm NAS when the vector $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_n\}$ is applied, we can show the following bound. Bounds of similar flavor have been frequently used in the scheduling literature (e.g. [9, 11, 14]).

Lemma 2.

$$C_j^{\boldsymbol{\alpha}} \leq t_j(\alpha_j) + \sum_{k:\alpha_k \leq \eta_k(\alpha_j)} \frac{p_k}{m} + (1 - \frac{1}{m}) p_j.$$

Proof. The completion time of job j is equal to the time to enter the queue for the parallel machines plus the waiting time in queue plus the processing time of job j.

The time to enter the queue is $t_j(\alpha_j)$, which is the α_j-point of job j in the single virtual machine that is m-times faster.

The wait time in the queue can be bounded as follows: Consider all jobs that entered the queue before job j, i.e., jobs belonging to the set $\{k \neq j : \alpha_k \leq \eta_k(\alpha_j)\}$ (which are all available for processing at time $t_j(\alpha_j)$ or earlier). Then the total time that needs to be processed before job j in the m machines is at most $\sum_{k\neq j:\alpha_k\leq\eta_k(\alpha_j)} p_k$. Thus the first time that a machine will free up is at most:

$$t_j(\alpha_j) + \frac{\sum_{k\neq j:\alpha_k\leq\eta_k(\alpha_j)} p_k}{m} = t_j(\alpha_j) - \frac{p_j}{m} + \sum_{k:\alpha_k\leq\eta_k(\alpha_j)} \frac{p_k}{m},$$

which is obtained by averaging the processing times of all jobs before j. Adding up the previous term with the processing time p_j gives the result.

Our deterministic algorithm will perform best by taking a fixed value of α for all jobs: $\alpha_j = \alpha,\ \forall j$. The following theorem is the main result of this section. Its proof is an extension of the proof of Theorem 3.3 in [9] to the parallel machine case.

Theorem 1. *Algorithm NAS is* $\max\{1 + \frac{1}{\alpha}, 2 + \alpha\}$*-competitive. In particular, for* $\alpha = \frac{\sqrt{5}-1}{2}$*, the schedule is* $\left(\frac{3+\sqrt{5}}{2}\right)$*-competitive (*$\frac{3+\sqrt{5}}{2} < 2.6181$*).*

Proof. Consider a canonical set $S = \{1, \ldots, l\}$ for the fast single machine. Fix a job $j \in S$ and let $\eta_k = \eta_k(\alpha)$ represent the fraction of job k processed before $t_j(\alpha)$. By reordering the elements in S such that $t_1(\alpha) \leq \ldots \leq t_l(\alpha)$, we have that

$$t_j(\alpha) - r_{\min}(S) = \sum_{k\in S} \eta_k \frac{p_k}{m} \leq \sum_{k=j}^{l} \alpha \frac{p_k}{m} + \sum_{k=1}^{j-1} \frac{p_k}{m} = \frac{\alpha}{m} p(S) + \frac{(1-\alpha)}{m} \sum_{k=1}^{j-1} p_k. \quad (2)$$

Let C_j^α be the completion time of job j output by algorithm NAS. Define R to be the set of jobs such that $t_k(\alpha) < r_{\min}(S)$; note that $R \bigcap S = \emptyset$ and $R \bigcup \{1, \ldots, j\} = \{k : \alpha \leq \eta_k\}$. Thus, combining Lemma 2 with Equation (2) and then noting that $\alpha \frac{p(R)}{m} \leq r_{\min}(S)$, we get

$$\begin{aligned} C_j^\alpha &\leq r_{\min}(S) + \frac{\alpha}{m} p(S) + \frac{(1-\alpha)}{m} \sum_{k=1}^{j-1} p_k + \frac{1}{m} p(R) + \frac{1}{m} \sum_{k=1}^{j-1} p_k + p_j \\ &\leq (1 + \frac{1}{\alpha}) r_{\min}(S) + \frac{\alpha}{m} p(S) + \frac{(2-\alpha)}{m} \sum_{k=1}^{j-1} p_k + p_j. \end{aligned}$$

Multiplying by p_j and summing over S we get

$$\sum_{j\in S} p_j C_j^\alpha \leq (1 + \frac{1}{\alpha}) r_{\min}(S) p(S) + \frac{\alpha}{m} p(S)^2 + \frac{(2-\alpha)}{m} \sum_{j\in S} \sum_{k=1}^{j-1} p_j p_k + \sum_{j\in S} p_j^2.$$

Using the identity $\sum_{j\in S}\sum_{k=1}^{j-1} p_j p_k = \frac{1}{2}p(S)^2 - \frac{1}{2}\sum_{j\in S} p_j^2$ we obtain that for any canonical set S:

$$\begin{aligned}\sum_{j\in S} p_j C_j^\alpha &\le (1+\frac{1}{\alpha})r_{\min}(S)p(S) + (2+\alpha)\frac{p(S)^2}{2m} + \sum_{j\in S} p_j^2 \\ &\le \max\{1+\frac{1}{\alpha}, 2+\alpha\}\left(p(S)\left(r_{\min}(S) + \frac{p(S)}{2m}\right) + \frac{1}{2}\sum_{j\in S} p_j^2\right).\end{aligned}$$

Assume now that the jobs are ordered such that $w_1/p_1 \ge \cdots \ge w_n/p_n \ge w_{n+1}/p_{n+1} = 0$. Let us now bound the overall cost of the schedule using Equation (1) applied to instance I_m and the feasibility of $Z_R^m(I)$:

$$\begin{aligned}\sum_{j\in N} w_j C_j^\alpha &= \sum_{i=1}^{n}\left(\frac{w_i}{\frac{p_i}{m}} - \frac{w_{i+1}}{\frac{p_{i+1}}{m}}\right)\sum_{l=1}^{k(i)}\sum_{j\in S_l^i}\frac{p_j}{m}C_j^\alpha \\ &\le \max\{1+\frac{1}{\alpha}, 2+\alpha\}\sum_{i=1}^{n}\left(\frac{w_i}{\frac{p_i}{m}} - \frac{w_{i+1}}{\frac{p_{i+1}}{m}}\right)\cdot \\ &\quad \sum_{l=1}^{k(i)}\left(\frac{p(S_l^i)}{m}\left(r_{\min}(S_l^i) + \frac{p(S_l^i)}{2m}\right) + \frac{1}{2m}\sum_{j\in S_l^i} p_j^2\right) \\ &\le \max\{1+\frac{1}{\alpha}, 2+\alpha\}\sum_{i=1}^{n}\left(\frac{w_i}{\frac{p_i}{m}} - \frac{w_{i+1}}{\frac{p_{i+1}}{m}}\right)\cdot \\ &\quad \sum_{l=1}^{k(i)}\left(\sum_{j\in S_l^i}\frac{p_j}{m}M_j + \frac{1}{2m}\sum_{j\in S_l^i} p_j^2\right) \\ &= \max\{1+\frac{1}{\alpha}, 2+\alpha\}\sum_{i=1}^{n}\left(\frac{w_i}{\frac{p_i}{m}} - \frac{w_{i+1}}{\frac{p_{i+1}}{m}}\right)\sum_{l=1}^{k(i)}\sum_{j\in S_l^i}\frac{p_j}{m}\left(M_j + \frac{p_j}{2}\right).\end{aligned}$$

Here, M_j denotes the mean-busy-time of job j in the LP-schedule. Applying Equation (1) again it follows that the previous quantity equals

$$\max\{1+\frac{1}{\alpha}, 2+\alpha\}\sum_{j\in N} w_j\left(M_j + \frac{p_j}{2}\right) = \max\{1+\frac{1}{\alpha}, 2+\alpha\}\left(Z_R^m(I) + \frac{1}{2}\sum_{j\in N} w_j p_j\right),$$

and by Lemma 1 it follows that $\sum_{j\in N} w_j C_j^\alpha \le \max\{1+\frac{1}{\alpha}, 2+\alpha\} Z^m(I)$. Finally, we recall that $Z^m(I)$ is a lower bound on the optimal offline cost of $P|r_j|\sum_j w_j C_j$, and the proof is complete.

As in the single machine case, there are instances for which algorithm NAS gives a schedule with cost as much as twice the LP lower bound; see, for example, [9]. However, we do not know whether our analysis is tight.

4 A Randomized Online Algorithm for $P|r_j|\sum_j w_j C_j$

Consider the following algorithm, which we denote as *Non-preemptive α Scheduling Randomized* (NASR).

Algorithm NASR:

INPUT: A scheduling instance $I = \{(p_i, r_i, w_i),\ i \in N\}$ which is revealed online and a distribution f.

(1) Construct the preemptive LP-schedule on a single virtual machine m-times faster ($I \mapsto I_m$).
(2) Each α_j is taken identically and independently from distribution $f(\alpha)$.
(3) At job j's α_j-point $t_j(\alpha_j)$, it enters into a FIFO queue for the m machines (job j is then scheduled the first time a machine is available after all preceding jobs in the queue have started).

We start by proving that, by utilizing the uniform distribution, algorithm NASR is 2-competitive; this matches Schulz and Skutella's [15] bound with a list-scheduling algorithm. The proof is very similar to Theorem 3.4 in [9].

Theorem 2. *NASR is 2-competitive when $f(\alpha)$ is the uniform distribution on $(0, 1]$.*

Proof. Let $C_j^{\boldsymbol{\alpha}}$ be the completion time of job j in the schedule given by algorithm NASR. We apply Lemma 2 and first find a conditional expectation, holding α_j constant:

$$\begin{aligned}
\mathrm{E}[C_j^{\boldsymbol{\alpha}} \mid \alpha_j] &\le t_j(\alpha_j) + \sum_{k \neq j} \frac{p_k}{m} \int_0^{\eta_k(\alpha_j)} d\alpha_k + p_j \\
&= t_j(\alpha_j) + \sum_{k \neq j} \frac{p_k}{m} \eta_k(\alpha_j) + p_j \\
&\le 2(t_j(\alpha_j) + \frac{p_j}{2}).
\end{aligned}$$

This implies that

$$\mathrm{E}\left\{C_j^{\boldsymbol{\alpha}}\right\} \le \int_0^1 2(t_j(\alpha_j) + \frac{1}{2}p_j) d\alpha_j = 2(M_j + \frac{1}{2}p_j),$$

where M_j denotes the mean-busy-time of job j in the LP-schedule. Multiplying by w_j and summing over j we get

$$\mathrm{E}\left[\sum_{j \in N} w_j C_j^{\boldsymbol{\alpha}}\right] \le 2\left(Z_R(I_m) + \frac{1}{2}\sum_{i \in N} w_i p_i\right) \le 2 \cdot Z^m(I),$$

which proves the result.

We now turn to deriving improved bounds which will depend on the number of machines. We show that by taking the α_j from an appropriate distribution we can improve on 2-competitiveness. To begin, we must first give a refinement of Lemma 2.

Lemma 3.

$$C_j^{\boldsymbol{\alpha}} \leq t_j(\alpha_j) + \sum_{\substack{k:\alpha_k \leq \eta_k(\alpha_j) \\ k \neq j}} \left(1 + \frac{\alpha_k}{m} - \frac{\eta_k(\alpha_j)}{m}\right) \frac{p_k}{m} + p_j.$$

Proof. As in Lemma 2, the completion time of job j is equal to the time to enter the queue for the parallel machines plus the wait-time in queue plus the processing time of job j. The only difference in the bound we are attempting to prove here lies in the in-queue waiting time. This can be bounded as follows.

Consider the set K of jobs that entered the queue before job j; i.e., $K = \{k : \alpha_k \leq \eta_k(\alpha_j), k \neq j\}$. If at time $t > t_k(\alpha_k)$ the fast machine is processing job k, then at least one of the parallel machines is busy (maybe processing a different job). Thus, at time $t_j(\alpha_j)$, the parallel machines have together processed $\sum_{k \in K} (\eta_k(\alpha_j) - \alpha_k) \frac{p_k}{m}$. Now, the total processing requirement entered into the queue before job j is $\sum_{k \in K} p_k$. Since we have just argued that by time $t_j(\alpha_j)$, the m machines have processed $\sum_{k \in K} (\eta_k(\alpha_j) - \alpha_k) \frac{p_k}{m}$, the remaining processing requirement in the system at time $t_j(\alpha_j)$ is

$$\sum_{k \in K} p_k - \sum_{k \in K} (\eta_k(\alpha_j) - \alpha_k) \frac{p_k}{m} = \sum_{k \in K} \left(1 + \frac{\alpha_k}{m} - \frac{\eta_k(\alpha_j)}{m}\right) p_k.$$

Using standard averaging arguments, the first time a machine will empty up to process job j is at most $\sum_{k \in K} \left(1 + \frac{\alpha_k}{m} - \frac{\eta_k(\alpha_j)}{m}\right) \frac{p_k}{m}$, and the proof is complete.

For a given job j, we partition $N \setminus \{j\}$ into N_1 and N_2. N_2 is the set of all jobs that are processed between the start and completion of job j on the fast virtual machine and N_1 consists of any remaining jobs. For any $k \in N_2$, we let μ_k denote the fraction of job j that, in the LP schedule of I_m, is processed before the start of job k. This implies $\forall k \in N_2$

$$\eta_k(\alpha_j) = \begin{cases} 0, & \alpha_j \leq \mu_k \\ 1, & \alpha_j > \mu_k. \end{cases}$$

Letting $t_j(0^+)$ denote the start time of job j, we may then write

$$t_j(\alpha_j) = t_j(0^+) + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} \frac{p_k}{m} + \alpha_j \frac{p_j}{m}.$$

Recalling that in the LP-schedule $M_j = \int_0^1 t_j(\alpha) d\alpha$ we have that

$$M_j = t_j(0^+) + \sum_{k \in N_2} (1 - \mu_k) \frac{p_k}{m} + \frac{1}{2} \frac{p_j}{m}. \tag{3}$$

We can now rewrite Lemma 3 as

$$C_j^{\boldsymbol{\alpha}} \leq t_j(0^+) + \sum_{\substack{k \in N_1 \\ \alpha_k \leq \eta_k(\alpha_j)}} \left(1 + \frac{\alpha_k}{m} - \frac{\eta_k(\alpha_j)}{m}\right) \frac{p_k}{m} + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} \left(2 - \frac{1-\alpha_k}{m}\right) \frac{p_k}{m} + \left(1 + \frac{\alpha_j}{m}\right) p_j.$$

This bound is useful in the proof of the main result in this section.

For any $m \geq 1$, consider the following equation, which extends the equation in Theorem 3.9 in [9] to an arbitrary number of machines:

$$\ln\left(1 + \tfrac{1}{m} - \tfrac{\gamma}{m}\right) + \tfrac{\gamma}{m} = \frac{e^{(-\gamma/m)}\left(1 + \frac{1}{m} - \frac{\gamma}{m} - e^{(-\gamma/m)}\right)\left(me^{(\gamma/m)} - \gamma e^{(\gamma/m)} + \frac{1}{m}e^{(\gamma/m)} + 1 - m\right)}{1 + \frac{1}{m} - \frac{\gamma}{m}}. \tag{4}$$

For any finite value of m, it can be shown that equation (4) has a unique solution $\gamma \in (0,1)$. We set

$$\frac{\delta_m}{m} = \ln\left(1 + \frac{1}{m} - \frac{\gamma}{m}\right) + \frac{\gamma}{m},$$

for the unique value of γ that satisfies equation (4). It can also be shown that $\delta_m \in (0,1)$ for any finite m. With this, we can consider the following distribution:

$$f(\alpha) = \begin{cases} c_m e^{(\alpha/m)}, & 0 \leq \alpha \leq \delta_m \\ 0, & o.w. \end{cases}$$

where $c_m = \left(m\left(e^{\delta_m/m} - 1\right)\right)^{-1}$. The main result is then the following.

Theorem 3. *With $f(\alpha)$ as above, NASR is $(1 + c_m)$-competitive.*

Proof. Omitted for space limitations.

The class of distributions we applied is optimal for our analysis. Essentially, equation (4) is a sufficient optimality condition for our distributions and analysis technique.

We can also prove that $c_m < 1$ for any finite $m \geq 1$ and $\lim_{m\to\infty} c_m = 1$. Not surprisingly then, as m grows, f uniformly approaches the uniform distribution on $(0,1]$. Figure 1 plots $1 + c_m$ as a function of the number of machines.

5 A Randomized Online Algorithm for $P|r_j, pmtn| \sum_j w_j C_j$

We now consider the simpler preemptive case. Consider the following algorithm *Preemptive α Scheduling Randomized* (PASR):

Algorithm PASR:

INPUT: A scheduling instance $I = \{(p_i, r_i, w_i),\ i \in N\}$ which is revealed online and a distribution f.

(1) Construct the preemptive LP-schedule on a single virtual machine m-times faster ($I \mapsto I_m$).
(2) Draw α randomly from the distribution $f(\alpha)$.
(3) Apply preemptive list-scheduling in order of non-decreasing $t_j(\alpha)$ on the m machines.

Repeating an observation from Schulz and Skutella [16], we see that at any given time, the *order* of the $t_j(\alpha)$ of already released jobs can be found, even if the actual values of $t_j(\alpha)$ are not known. It is interesting to note that if step (2) is replaced by "Take $\alpha = 1$" the algorithm becomes a deterministic online algorithm and it coincides with Megow and Schulz's 2-competitive algorithm for $P|r_j, pmtn| \sum_j w_j C_j$ [13]. On the other hand, if f is taken as the uniform distribution in $[0, 1]$, PASR is also 2-competitive (and this follows as a consequence of the forthcoming analysis).

Let C_j^α denote the completion time of job j in the schedule output by algorithm PASR. Consider job j. Define J as the set of jobs that start before job j in the LP-schedule. For any $k \neq j$, let η_k denote the fraction of job k that is completed in the LP-schedule by time $t_j(0^+)$. Note that $\eta_k = 0,\ \forall k \notin J$. We also have that $t_j(0^+) \geq \sum_{k \in J} \eta_k \frac{p_k}{m}$. Now, define $K_1 = \{k \mid t_k(\alpha) < t_j(0^+)\}$ and $K_2 = \{k \mid t_j(0^+) < t_k(\alpha) < t_j(\alpha)\}$. So $K = K_1 \cup K_2$ is the set of jobs that can preempt job j. Note that jobs $k \in K_2$ preempt job j in the LP-schedule and are all processed in the interval $[t_j(0^+), t_j(\alpha)]$. Consequently, $t_j(\alpha) = t_j(0^+) + \sum_{k \in K_2} \frac{p_k}{m} + \alpha \frac{p_j}{m}$. The following bound is central to our analysis.

Lemma 4.

$$C_j^\alpha \leq t_j(\alpha) + (1 - \frac{\alpha}{m}) p_j + \sum_{k \in J, \eta_k \geq \alpha} (1 - \frac{\eta_k}{m}) \frac{p_k}{m}.$$

Proof. We first note that if the LP-schedule is busy, then at least one machine is busy in the schedule defined by PASR. Thus, by time $t_j(0^+)$, the LP-schedule will have processed a total of $\sum_{k \in K_1} \eta_k \frac{p_k}{m}$ and consequently, so will have the schedule defined by algorithm PASR.

We now make some assumptions that can only increase the completion time of job j: (1) Job j has not begun processing in the schedule defined by PASR at time $t_j(0^+)$ and (2) jobs $k \in K_2$ are released at time $t_j(0^+)$ (note that, originally, jobs in K_2 were released sometime in the interval $[t_j(0^+), t_j(\alpha)]$). While it is obvious that (1) will only increase the completion time of job j, it is not immediately clear that (2) also increases the completion time of job j. However, as we are dealing with preemptive scheduling and as job j cannot finish before $t_j(\alpha)$, making jobs in K_2 available earlier only increases the times $t_j(x)$ for any $0 \leq x \leq 1$ (since it was possible that job j found some open machine

in the interval $[t_j(0^+), t_j(\alpha)]$). Thus, job j's completion time can only increase under assumption (2).

Thus, under Assumptions (1) and (2), at time $t_j(0^+)$, the amount of *available* processing that remains from $K_1 \cup K_2$ is at most $\sum_{k\in K_2} p_k + \sum_{k\in K_1}(1-\frac{\eta_k}{m})p_k$. Since we have m machines, by standard averaging arguments, we have that

$$\begin{aligned} C_j^\alpha &\le t_j(0^+) + \sum_{k\in K_2}\frac{p_k}{m} + \sum_{k\in K_1}(1-\frac{\eta_k}{m})\frac{p_k}{m} + p_j \\ &= t_j(0^+) + \sum_{k\in K_2}\frac{p_k}{m} + \sum_{k\in J,\eta_k\ge\alpha}(1-\frac{\eta_k}{m})\frac{p_k}{m} + p_j \\ &= t_j(0^+) + \sum_{k\in K_2}\frac{p_k}{m} + \alpha\frac{p_j}{m} + (1-\frac{\alpha}{m})p_j + \sum_{k\in J,\eta_k\ge\alpha}(1-\frac{\eta_k}{m})\frac{p_k}{m} \\ &= t_j(\alpha) + (1-\frac{\alpha}{m})p_j + \sum_{k\in J,\eta_k\ge\alpha}(1-\frac{\eta_k}{m})\frac{p_k}{m}, \end{aligned}$$

and the proof is complete.

The next lemma generalizes a result by Schulz and Skutella [16]; its proof is omitted for space limitations.

Lemma 5. *Suppose there exists a distribution $f(\alpha)$ and a constant $\gamma_m \in (0,1)$ such that:*

- $\max_{\alpha\in[0,1]} f(\alpha) \le 1+\gamma_m$.
- $(1-\frac{\eta}{m})\int_0^\eta f(\alpha)d\alpha \le \gamma_m\eta,\ \forall\eta\in[0,1]$.
- $1-\frac{\mathrm{E}\{\alpha\}}{m} \le \frac{1+\gamma_m}{2}$.

Then, Algorithm PASR is $(1+\gamma_m)$-competitive.

Consider the following distribution for α for the case where we have m machines:

$$f(\alpha) = \begin{cases} \gamma_m\frac{m^2}{(m-\alpha)^2}, & \alpha\in[0,\delta_m] \\ (1+\gamma_m) & \alpha\in(\delta_m,1], \end{cases}$$

where $\gamma_m = \frac{\delta_m(m-\delta_m)}{m-\delta_m(1-\delta_m)}$ and $\delta_m \in (0,1]$.

Note that for $m=1$, if we let $\delta_1 = \frac{1}{2}$, then $\gamma_1 = \frac{1}{3}$ and f is exactly the distribution chosen by Schulz and Skutella [16], which gives a $(1+\gamma_1) = \frac{4}{3}$-competitive algorithm. We are able to find an optimal value δ_m^* for all $m \ge 2$ and show that the above distribution satisfies the conditions of Lemma 5, proving the main result of this section.

Theorem 4. *Algorithm PASR is $(1+\gamma_m)$-competitive for $m \ge 2$.*

Proof. Omitted for space limitations.

Figure 1 plots $1+\gamma_m$ as a function of the number of machines.

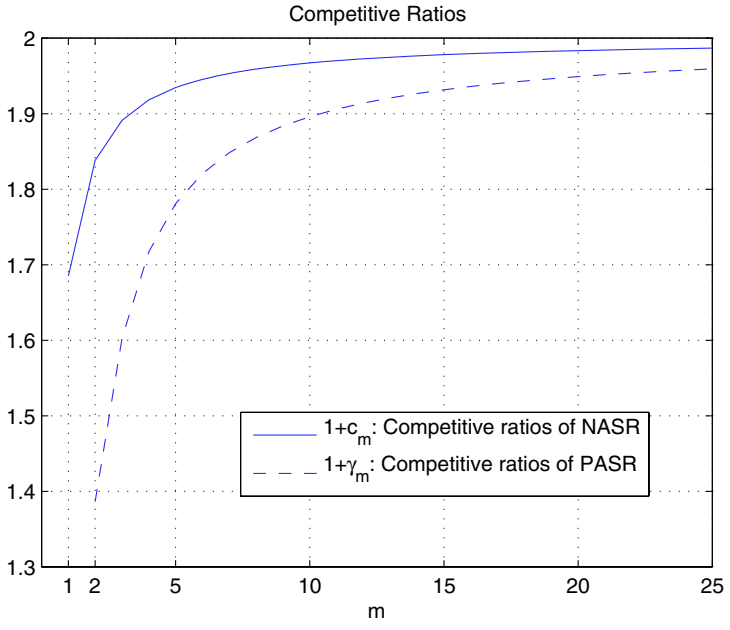


Fig. 1. The competitive ratios $1 + c_m$ and $1 + \ _m$ as a function of

Acknowledgments

We thank Andreas Schulz for carefully reading this manuscript and for several pointers to relevant literature. The research of the first author was supported by postdoctoral grant MECESUP UCH 0109. The research of the second author was supported by an MIT ORC research assistantship from Professor Patrick Jaillet.

References

1. Afrati, F., Bampis, E., Chekuri, C., Karger, D., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., Sviridenko, M.: Approximation schemes for minimizing average weighted completion time with release dates. Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS) (1999) 32–43
2. Anderson, E.J., Potts, C.N.: On-line scheduling of a single machine to minimize total weighted completion time. Mathematics of Operations Research **29** (2004) 686–697
3. Chakrabarti, S., Phillips, C., Schulz, A.S., Shmoys, D.B., Stein, C., Wein, J.: Improved scheduling algorithms for minsum criteria. In: Automata, Languages and Programming (ICALP), Springer Lecture Notes in Computer Science **1099** (1996) 646–657

4. Chekuri, C., Motwani, R., Natarajan B., Stein, C.: Approximation techniques for average completion time scheduling. SIAM Journal on Computing **31** (2001) 146–166
5. Chou, C.M., Queyranne M., Simchi-Levi, D.: The asymptotic performance ratio of an on-line algorithm for uniform parallel machine scheduling with release dates. In: Proceedings of the 8th Integer Programming and Combinatorial Optimization Conference (IPCO), Springer Lecture Notes in Computer Science **2081** (2001) 45–59
6. Eastman, W.L., Even, S., Isaacs, I.M.: Bounds for the optimal scheduling of jobs on processors. Management Science **11** (1964) 268–279
7. Goemans, M.X.: A supermodular relaxation for scheduling with release dates. In: Proceedings of the 5th Integer Programming and Combinatorial Optimization Conference (IPCO), Springer Lecture Notes in Computer Science **1084** (1996) 288–300
8. Goemans, M.X.: Improved approximation algorithms for scheduling with release dates. In: Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (1997) 591–598
9. Goemans, M.X., Queyranne, M., Schulz, A.S., Skutella, M., Wang, Y.: Single machine scheduling with release dates. SIAM Journal on Discrete Mathematics **15** (2002) 165–192
10. Graham, R.L., Lawler, E.L., Lenstra J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics **5** (1979) 287–326
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** (1997) 513–544
12. Hoogeveen, J.A., Vestjens, A.P.A.: Optimal on-line algorithms for single-machine scheduling. In: Proceedings of the 5th Integer Programming and Combinatorial Optimization Conference (IPCO), Springer Lecture Notes in Computer Science **1084** (1996) 404–414
13. Megow N., Schulz, A.S.: On-line scheduling to minimize average completion time revisited. Operations Research Letters **32** (2004) 485–490
14. Phillips, C., Stein C., Wein, J.: Minimizing average completion time in the presence of release dates. Mathematical Programming **82** (1998) 199–223
15. Schulz, A.S., Skutella, M.: Scheduling unrelated machines by randomized rounding. SIAM Journal on Discrete Mathematics **15** (2002) 450–469
16. Schulz, A.S., Skutella, M.: The power of -points in preemptive single machine scheduling. Journal of Scheduling **5** (2002) 121–133
17. Seiden, S.: A guessing game and randomized online algorithms. In: Proceedings of the 32nd ACM Symposium on Theory of Computing (2000) 592–601
18. Sitters, R.: Complexity and approximation in routing and scheduling. Ph.D. Thesis, Eindhoven University of Technology, Netherlands (2004)
19. Stougie, L., Vestjens, A.P.A.: Randomized algorithms for on-line scheduling problems: how low can't you go?. Operations Research Letters **30** (2002) 89–96
20. Vestjens, A.P.A.: Online machine scheduling. Ph.D. Thesis, Eindhoven University of Technology, Netherlands (1997)

Unique Sink Orientations of Grids*

Bernd Gärtner[1], Walter D. Morris[2], and Leo Rüst[3]

[1] Institute of Theoretical Computer Science, ETH Zürich,
CH-8092 Zürich, Switzerland
gaertner@inf.ethz.ch
[2] Department of Mathematical Sciences, George Mason University,
MS 3F2, US-Fairfax VA 22030-4444, USA
wmorris@gmu.edu
[3] Institute of Theoretical Computer Science, ETH Zürich,
CH-8092 Zürich, Switzerland
ruestle@inf.ethz.ch

Abstract. We introduce unique sink orientations of grids as digraph models for many well-studied problems, including linear programming over products of simplices and generalized linear complementarity problems over P-matrices (PGLCP). We investigate the combinatorial structure of such orientations and develop randomized algorithms for finding the sink. We show that the orientations arising from PGLCP satisfy the combinatorial *Holt-Klee* condition known to hold for polytope digraphs, and we give the first expected linear-time algorithms for solving PGLCP with a fixed number of blocks.

1 Introduction

A *grid* is a graph whose vertex set is the Cartesian product of n finite sets, with edges joining all pairs of vertices that differ in exactly one component.

If all sets have size two, we get the graph of the n-cube. A *face* or *subgrid* is any induced subgraph spanned by the Cartesian product of subsets of the original sets. An orientation ψ of the grid is called a *unique sink orientation* (USO) if any nonempty face has a unique sink with respect to ψ. Figure 1 (left) depicts a USO of the $(3 \times 2 \times 2)$-grid. In particular, the grid itself must have a unique global sink. Grid USO may contain directed cycles, as the 3-cube in Figure 1 (right) shows.

The significance of USO on grids comes from the fact that they form a simple combinatorial framework subsuming a number of well-studied problems. We show in this paper that the problem of solving a *generalized linear complementarity problem* over a P-matrix (PGLCP), as introduced by Cottle and Dantzig

* The first and the third author acknowledge support from the Swiss Science Foundation (SNF), Project No. 200021-100316/1. Part of this research was done at the 2004 Barbados Undercurrent Workshop *Polyhedra, Convex Geometry, and Optimization* at Bellairs Research Institute, McGill University. For a full paper see [1].

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 210–224, 2005.

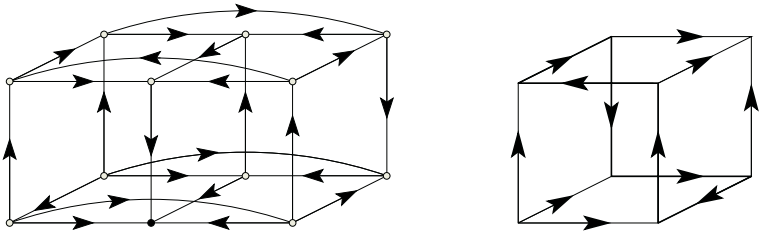


Fig. 1. Left: USO of the $(3 \times 2 \times 2)$-grid. Right: cyclic USO of the 3-cube

[2], can be recast as the problem of finding the unique sink of an implicitly given grid USO. As special cases, this includes the well-known standard linear complementarity problems over P-matrices (PLCP) [3], linear programming (LP) over products of simplices, and LP over combinatorial cubes. In the LP applications, we get acyclic unique sink orientations (AUSO).

Two major open problems motivate our research. On the one hand, it is unknown whether polynomial-time algorithms exist for PLCP or PGLCP, even though both problems are unlikely to be NP-hard. Megiddo has shown that hardness of PLCP would imply NP = co-NP [4], and his proof extends to PGLCP easily. LP, on the other hand, is solvable in polynomial time (a celebrated result of Khachyian [5]), but a *strongly* polynomial algorithm is not known, even if we are dealing only with LP over combinatorial cubes. Candidates for strongly polynomial algorithms must be *combinatorial* in the sense that the number of arithmetic operations they perform depends only on the combinatorial structure of the LP but not on the actual numbers that encode it. The AUSO approach attempts to extract the combinatorial structure behind LP; in this paper, we generalize to the combinatorics of PGLCP. A polynomial-time algorithm for finding the sink of a grid USO (using an oracle that returns the orientation of a given edge) would solve both problems in strongly polynomial time. It seems unlikely that such an algorithm for general USO will be easier to find than one for PGLCP. Still, the generalization reveals some (algorithmically useful) hidden structure, leading to new results for PGLCP. Ultimately, results obtained along these lines may help to resolve the (combinatorial) complexities of LP and PGLCP.

The AUSO framework also covers a generalization of LP resulting from the replacement of linear objective functions with *abstract objective functions* (AOF) [6, 7], or *completely unimodal numberings* of vertices [8, 9]. On general polytopes, these concepts are dual to the notion of *shellings* [9], and they have successfully been applied to the theory of polytope (di)graphs and linear programming [10, 11].

The case of grid AUSO—equivalently, AOF on products of simplices—has been treated in detail by Björklund et al. as a combinatorial framework for the problem of computing optimal infinite game strategies [12, 13]. The games considered include parity, mean-payoff, and simple stochastic games. Whether parity games (the easiest among the three) can be solved in polynomial time is an important open question [14].

In the planar case ($n = 2$), the combinatorial and algorithmic properties of AUSO have been examined by Tschirschnitz et. al. [15, 16].

Unique sink orientations of cubes (not necessarily acyclic) were first considered by Stickney and Watson as digraph models for PLCP [17]. Most remarkably, Szabó and Welzl gave algorithms for finding the sink of an n-cube USO by looking at only $O(c^n)$ vertices and edges, for some c strictly smaller than 2 [18]. This in particular yields the first combinatorial algorithms for PLCP with nontrivial runtime bounds. Unique sink orientations of the graphs of general polytopes are dual to *exact signings* studied by Kleinschmidt and Onn [19].

In this paper, we generalize results known to hold in some of the above special cases, and we prove new structural and algorithmic results of particular significance for the theory of PLCP and PGLCP. Probably the most surprising fact is that all these results hold even under the presence of directed cycles in the orientation.

We develop two simple *randomized* algorithms whose expected number of calls to the oracle is of the order $f(n)N$, with N being the sum of sizes of the n sets whose product forms the grid, and with $f(n) \approx n!$. If n is fixed, we get *linear-time* algorithms. Specialized to PGLCP, this corresponds to the case in which we have a fixed number of *blocks*; in this situation, we get the first algorithms whose expected complexity is linear in the number of variables—this is optimal.

In the acyclic case, linear-time algorithms for fixed n are known. This follows from the fact that the problem of finding the sink in an AUSO can be formulated as an *LP-type problem* [20] in a natural way [12]. In the LP-type framework, a number of $f(n)N$ oracle calls suffices, even deterministically [21]. The currently best algorithm for the acyclic case combines two randomized algorithms [20, 22] and requires an expected number of $O(Nn + f(n))$ oracle calls, where $f(n) = \exp(O(\sqrt{n \log n}))$ is a *subexponential* function [23]. No subexponential bounds are known for USO that contain cycles.

Unique sink orientations coming from LP over products of simplices satisfy an interesting *geometric* property: consider any subgrid along with its unique sink and unique source.[1] The *Holt-Klee* (HK) condition states that there is a set of as many vertex-disjoint directed paths between source and sink as there are neighbors of the source (equivalently, the sink) in the subgrid [24]. The HK condition does not hold for general AUSO; there are two nonequivalent AUSO of the 3-cube with no set of three vertex-disjoint directed paths from source to sink. The HK condition is important as the only known simple combinatorial condition that can distinguish geometric from abstract situations, and there is an interesting algorithmic scenario in which this distinction becomes apparent [25, 26].

We prove that all grid USO coming from PGLCP do satisfy the HK condition (we also say that they are HK), even in the presence of cycles. This result emphasizes the geometric nature of PGLCP and establishes a new combinatorial

[1] The existence of a unique source follows from Theorem 1.

way of proving that a given USO cannot be realized as a PGLCP instance. The result is new also in the context of PLCP with its wide range of applications [3, 27].

The rest of this paper is organized as follows. After giving some basic results in the next section, Section 3 introduces two problems giving rise to USO of grids. Section 4 analyses two algorithms finding the sink of a grid USO and the final section is devoted to the proof that PGLCP induced orientations fulfill the HK condition.

2 Basics

Throughout this paper, we fix two natural numbers $N \geq n \geq 1$ and an ordered partition

$$\Pi = (\Pi_1, \ldots, \Pi_n)$$

of the set $[N] := \{1, \ldots, N\}$ into n nonempty subsets. We also refer to Π_i as the *block* i.

A subset $J \subseteq [N]$ is called a *Π-vertex* (or simply *vertex*) if $|J \cap \Pi_i| = 1$ for all i. Let V be the set of all vertices. The n-dimensional *grid* spanned by $S \subseteq [N]$ is the undirected graph $G(S) = (V(S), E(S))$, with

$$V(S) := \{J \in V \mid J \subseteq S\}, \quad E(S) := \{\{J, J'\} \subseteq V(S) \mid |J \oplus J'| = 2\}.$$

The vertices of $G(S)$ canonically correspond to the elements of the Cartesian product

$$\prod_{i=1}^{n} S_i, \quad S_i := S \cap \Pi_i.$$

Edges join pairs of vertices J, J' that differ in exactly one coordinate.

A *face* or *subgrid* of $G(S)$ is any graph of the form $G(S')$, for $S' \subseteq S$. Throughout, we abbreviate $G([N])$ as G.

Definition 1. *Let ψ be an orientation of G. ψ is called a* unique sink orientation *(USO) if all nonempty faces of G have unique sinks w.r.t. ψ.*

If ψ induces the directed edge (J, J'), we also write $J \xrightarrow{\psi} J'$.

Outmap and h-Vector. Any USO can be specified by associating each vertex J with its outgoing edges. Given J and $j \in [N] \setminus J$, we define $J \rhd j$ to be the unique vertex $J' \subseteq J \cup \{j\}$ which is different from J, and we call J' the *neighbor* of J *in direction* j. Note that J is a neighbor of J' in some direction different from j.

Given an orientation ψ, the function $s_\psi : V \to 2^{[N]}$, defined via

$$s_\psi(J) := \{j \in [N] \setminus J \mid J \xrightarrow{\psi} J \rhd j\}, \tag{1}$$

is called the *outmap* of ψ. Björklund et. al consider the outmap for acyclic grid USO and call it VID-function (vector of improving directions) [12]. Szabó and

Welzl [18] deal with outmaps for cube USO; formally, these are different from ours, even when we specialize to the cube case, because Szabó and Welzl identify the n-cube vertices with the subsets of some n-element set, while we identify them with certain n-subsets of some $2n$-element set. Still, we can generalize the characterization of USO outmaps by Szabó and Welzl, using one more ingredient.

Definition 2. *Let $s : V \to 2^{[N]}$. The vector $h(s) = (h_0(s), \ldots, h_{N-n}(s))$, defined via*

$$h_k(s) = \#\{J \in V \mid |s(J)| = k\}$$

is called the h-vector *of s. If $s = s_\psi$ for some orientation ψ on G, we also refer to $h(s)$ as $h(\psi)$.*

The following result is well-known for the acyclic case [9, 12] but going through its proof, one realizes that only the unique sink property is used [15].

Theorem 1. *$h(\psi) = h(\psi')$ for any two USO ψ, ψ' of G.*

As all USO on a grid have the same h-vector, we denote it by $h(\Pi)$, emphasizing the fact that it depends on the parameters of the grid only. An immediate corollary of this theorem is that the h-vector is symmetric, meaning that $h_k(\Pi) = h_{N-n-k}(\Pi)$ for all $k \in \{0, \ldots, N-n\}$ (reversing all edge orientations of a USO yields a USO again [1]). In particular, w.r.t. any given USO, all nonempty subgrids of G also have unique *sources*.

Here is the characterization of functions s that are of the form s_ψ, for ψ being USO (see [1] for a proof).

Lemma 1. *Let $s : V \to 2^{[N]}$ satisfy $s(J) \cap J = \emptyset$ for all $J \in V$. Such an s is the outmap of a USO of G if and only if*

(i) $(s(J) \oplus s(J')) \cap (J \oplus J') \neq \emptyset$, *for all $J \neq J'$, and*
(ii) $h(s) = h(\Pi)$.

Refined Index. The outmap value $s_\psi(J) \subseteq [N]$ of a vertex J w.r.t. some USO ψ is partitioned according to the dimensions of the grid (the sets Π_i). Accordingly, the outdegree of J can be refined to an n-vector of dimensional outdegrees, as follows.

Definition 3. *Let ψ be a grid USO. The function*

$$r_\psi : V \to \prod_{i=1}^{n} \{0, \ldots, |\Pi_i| - 1\},$$

with

$$r_\psi(J) = \left(|s_\psi(J) \cap \Pi_1|, \ldots, |s_\psi(J) \cap \Pi_n|\right), \quad J \in V$$

is called the refined index *of ψ.*

Unlike the outmap, the refined index is a mapping between two sets of the same size, so it is natural to ask whether this mapping is a bijection. This is true in the cube case (where the refined index is just a different way of writing the outmap) [18], and it also holds for grid AUSO [12], where Björklund et al. use the term *signature* (SIG). The proof of the latter result does not generalize to the case of general USO. We show in [1] that the refined index is a bijection in general.

Theorem 2. *Let ψ be a grid USO. The refined index r_ψ is a bijection.*

In the proof, we make use of the concept of *inherited orientations*. Given a grid orientation, an inherited grid orientation can be obtained by collapsing dimensions of the grid, merging vertices along the collapsed dimensions to hypervertices. Hypervertices correspond to faces of the original grid, and the outmap of a hypervertex is defined as the outmap of the corresponding face sink.

The important fact is that inherited orientations of USO are USO again. This is utilized to get a contradiction in the case of two vertices with the same refined index: collapse the grid until we get a 1-dimensional grid USO where two vertices have the same outdegree, contradicting Theorem 1.

The bijection property of the refined index is useful in the proof of the following theorem [1]. Previously, this was only known to hold for USO of 2-dimensional grids that satisfy the Holt-Klee condition [15, 16].

Theorem 3. *Any unique sink orientation of a 2-dimensional grid is acyclic.*

3 Grid LP and Generalized LCP

In this section, we present two geometric models of grid USO arising from *Grid LP* and *Generalized LCP*. Consider a linear program in the variables $x = (x_1, \ldots, x_N)^T$, of the form

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} && Ax = b, \\ & && x \geq 0, \end{aligned} \tag{2}$$

where $A \in \mathbb{R}^{n \times N}, b \in \mathbb{R}^n, c \in \mathbb{R}^N$. For $S \subseteq [N]$, let A_S denote the submatrix of columns indexed by S. Furthermore assume that every vertex J is a nondegenerate basis in (2), meaning that $A_J^{-1} b > 0$. We say that the LP is *Π-compatible* (or, equivalently, the LP is a *Grid LP*), and we call A_J a *representative submatrix*. We will assume that the ordering of the columns in A_J is compatible with Π, meaning that the i-th column of A_J comes from block $\Pi_i, i \in [n]$.

The feasible region of a Π-compatible LP is combinatorially equivalent to the product of n simplices with a total of N facets. A unique sink orientation of the grid G is obtained from a Π-compatible LP with *generic* objective function vector c, meaning that $c^T x$ is not constant on any edge. In this case, an edge can be directed towards its vertex of lower objective function value. The resulting

orientation is a USO, with the unique sink of the face $G(S)$ corresponding to the unique optimal basis of the linear program resulting from (2) by restricting to the variables with indices in S.

The simplex algorithm (Chvátal's book [28] contains an excellent introduction) determines the edge orientations in terms of *reduced cost coefficients*. More precisely, if J is some basis, the row vector

$$\bar{c}(J) := c^T - c_J^T A_J^{-1} A \tag{3}$$

is the *reduced cost vector* associated with J. Note that $\bar{c}(J)_J = 0$, and that $\bar{c}(J)_j \neq 0$ for $j \notin J$, if c is generic. Thus, if J' is adjacent to J, with j being the unique index in $J' \setminus J$, we have

$$J \to J' \quad \Leftrightarrow \quad \bar{c}(J)_j < 0. \tag{4}$$

The existence of a unique sink therefore provides us with a solution to the following feasibility problem.

Definition 4. *Let $A \in \mathbb{R}^{n \times N}$ and $c \in \mathbb{R}^N$ such that A has* property P*, meaning that all determinants of representative submatrices A_J of A have the same nonzero sign. The* P-generalized linear complementarity problem (PGLCP) *defined by (A, c) is the problem of finding a vector $y \in \mathbb{R}^n$ such that $c^T \geq y^T A$, and with the property that for every $i \in [n]$, there is some $j \in \Pi_i$ satisfying $c_j = (y^T A)_j$.*

Intuitively, PGLCP is LP of the form (2) 'without a right-hand side'. It turns out that under this generalization, uniqueness of solution as well as the USO formulation persist. Because already the cyclic cube USO of Figure 1 (right) arises from a PGLCP [17], the generalization is proper.

Theorem 4. *Let $A \in \mathbb{R}^{n \times N}$ and $c \in \mathbb{R}^N$ define a* PGLCP *instance. Then*

(i) there exists a unique solution $y \in \mathbb{R}^n$ to (A, c), and
(ii) if c is generic, the edge orientations given by (4) define a unique sink orientation of the grid G.

Proof. Fix some vertex J, define

$$\begin{aligned} M^T &:= A_J^{-1} A, \\ q^T &:= \bar{c}(J) = c^T - c_J^T A_J^{-1} A, \end{aligned}$$

and consider the problem of finding $z \in \mathbb{R}^n, w \in \mathbb{R}^N$ such that

$$w - Mz = q, \tag{5}$$

$$\prod_{j \in \Pi_i} w_j = 0, \quad i \in [n], \tag{6}$$

and

$$w, z \geq 0. \tag{7}$$

By definition, every solution must satisfy

$$z_i = w_j, \quad i \in [n], j \in J_i, \tag{8}$$

meaning that (6) is equivalent to

$$z_i \prod_{j \in \Pi_i} w_j = 0. \tag{9}$$

Equations (5), (7) and (9) define the P-generalized linear complementarity problem according to Cottle & Dantzig who show that a feasible solution (w, z) exists if every representative submatrix of M^T is a P-*matrix* [2]. A matrix is a P-matrix if all determinants of principal minors are positive [3]. In our case, M^T satisfies this, which easily follows from the fact that M^T has property P *and* contains a representative identity matrix. Note that the n variables $w_j, j \in J$ of system (5) are redundant by (8) and can be deleted from the problem, along with their corresponding rows.

To prove the existence of y in part (i), it remains to observe that y fulfills the conditions of Definition 4 if and only if $z^T = c_J^T - y^T A_J$ and $w^T = c^T - y^T A$ solve (5), (6) and (7). The uniqueness of y follows from the known uniqueness proofs in the setup of Cottle & Dantzig [29, 30].

Statement (ii) is a corollary of (i), because c being generic implies that the vector y from (i) can be expressed in the form $y^T = c_K^T A_K^{-1}$ for exactly one K. This set K is the unique sink of G in the orientation defined by (4). The fact that this orientation defines a USO easily follows: applying the above arguments to the PGLCP instance (A_S, c_S), we can prove the existence of a unique sink in the subgrid $G(S)$. □

4 Algorithms

In this section, we develop randomized algorithms for finding the sink of a given grid USO, implicitly specified by an *edge evaluation* oracle. The oracle must be able to return the orientation of any given grid edge. Our complexity measure will be the maximum (expected) number of oracle calls needed to find the sink in the worst case. In all concrete instances, this oracle can easily be implemented in polynomial time, meaning that the number of oracle calls is a good measure of complexity. In the case of PGLCP, for example, an edge evaluation must return the sign of a single coefficient of the reduced cost vector (3).

Any USO algorithm which calls the oracle only a polynomial number of times is actually a *strongly* polynomial algorithm for LP and PGLCP. Moreover, the complexity of a single edge evaluation typically only depends on n but not on N (in the PGLCP case, this complexity is $O(n^3)$). Thus, if n is considered to be a constant, any bound on the number of edge evaluations determines the complexity of the algorithm up to a constant factor.

4.1 The Product Algorithm

This algorithm generalizes the *product algorithm* of Szabó and Welzl from cubes [18] to grids, with a slight twist: while in the n-cube, all dimensions are equivalent with respect to their size (which is two), a general grid may have 'heavy' dimensions (with large Π_i) and 'light' dimensions. Our algorithm gives priority to the heavy dimensions. Recall that $S_i := S \cap \Pi_i$. A generic call to PRODUCT finds the sink of a nonempty face $G(S)$, see Figure 2 (left).

Algorithm 5
PRODUCT(S):
  IF is_vertex(S) THEN
    RETURN S
  ELSE
    choose a heaviest $S_i \in S$
    choose $j \in S_i$ at random
    $K :=$ PRODUCT($S \setminus \{j\}$)
    $K' := K \triangleright j$
    IF $K' \xrightarrow{\psi} K$ THEN
      RETURN K
    ELSE
      RETURN
      PRODUCT($(S \setminus S_i) \cup \{j\}$)
    END
  END

Algorithm 6
RANDOMFACET(J, S):
  IF $S = J$ THEN
    RETURN J
  ELSE
    choose a heaviest $S_i \in S$
    choose $j \in S_i \setminus J_i$ at random
    $K :=$ RANDOMFACET($J, S \setminus \{j\}$)
    $K' := K \triangleright j$
    IF $K' \xrightarrow{\psi} K$ THEN
      RETURN K
    ELSE
      RETURN
      RANDOMFACET($K', (S \setminus S_i) \cup \{j\}$)
    END
  END

Fig. 2. The Algorithms PRODUCT and RANDOMFACET

The algorithm recursively computes the sink K of the subgrid $G(S \setminus \{j\})$. If the edge incident to K in direction j is incoming, we have already found the global sink, otherwise we need to search the lower-dimensional 'facet' $G((S \setminus S_i) \cup \{j\})$ recursively. To prepare the analysis of the algorithm, let $S \subseteq [N]$ be a set containing a vertex and let $z = \max_{i=1}^{n} |S_i|$ be the size of a heaviest dimension. The $(z-1)$-vector $(a_2, \ldots, a_z)$, defined through

$$a_t = |\{i \in [n] \mid |S_i| = t\}|$$

is called the *characteristic* of S. If $z = 1$ (meaning that S is a vertex itself), the characteristic is the empty vector $()$.

It can easily be shown by induction that the expected number of edge evaluations in PRODUCT(S) only depends on the characteristic of S but (maybe surprisingly) not on the input USO ψ. We can even compute the exact expectation.

Theorem 5. *For $z \geq 1$, let $T_e(a_2, \ldots, a_z)$ denote the expected number of edge evaluations in a call to* PRODUCT(S)*, where S has characteristic $(a_2, \ldots, a_z)$. Then*

$$\begin{aligned} T_e(a_2, \ldots, a_z) = {} & T_e(a_2, \ldots, a_{z-1} + 1, a_z - 1) + 1 \\ & + T_e(a_2, \ldots, a_{z-1}, a_z - 1)/z \end{aligned} \tag{10}$$

for $z > 1$, with $T_e(a_2, \ldots, a_{z-1}, 0) := T_e(a_2, \ldots, a_{z-1})$ and $T_e() = 0$. The solution to this recurrence is

$$T_e(a_2, \ldots, a_z) = \prod_{k=2}^{z} H_k^{a_k} + \sum_{k=2}^{z} \prod_{\ell=k}^{z} (H_\ell - H_{k-1} + 1)^{a_\ell} - z. \tag{11}$$

Here, H_k is the k-th Harmonic number.

Proof. It is clear that $(a_2, \ldots, a_{z-1} + 1, a_z - 1)$ and $(a_2, \ldots, a_{z-1}, a_z - 1)$ are the characteristics of the grids handled in the recursive calls. Moreover, the second recursive call is executed if and only if the global sink contains the chosen element j. This happens with probability $1/z$. The recurrence follows. The closed form (11) can be checked by induction. □

For fixed z, (11) is maximized if $a_z = n$. This corresponds to the characteristic $(0, \ldots, 0, n)$ of the $(z \times \cdots \times z)$-grid. It follows that

$$T_e(a_2, \ldots, a_z) \leq H_z^n + \sum_{k=2}^{z} (H_z - H_{k-1} + 1)^n - z. \tag{12}$$

The middle term

$$f(n, z) := \sum_{k=2}^{z} (H_z - H_{k-1} + 1)^n \tag{13}$$

asymptotically dominates the bound in (12), and an estimate of $f(n, z) \leq (z-1)H_z^n \approx z \ln^n z$ immediately follows. The next result shows that the bound is actually *linear* in z.

Lemma 2.

$$T_e(a_2, \ldots, a_z) \leq (\lfloor en! \rfloor - 1)\, z + H_z^n.$$

Proof. Using the estimate

$$\sum_{t=\ell}^{u} g(t) \leq \int_{\ell-1}^{u} g(x)\, dx \tag{14}$$

for any decreasing function g such that the integral exists, we can bound (13) as follows.

$$\begin{aligned} f(n, z) &= \sum_{k=2}^{z} (H_z - H_{k-1} + 1)^n \overset{(14)}{\leq} \sum_{k=2}^{z} (\ln z - \ln(k-1) + 1)^n \\ &= \sum_{k=1}^{z-1} \left(1 - \ln \frac{k}{z}\right)^n < \sum_{k=1}^{z} \left(1 - \ln \frac{k}{z}\right)^n \\ &\overset{(14)}{\leq} \int_0^z \left(1 - \ln \frac{k}{z}\right)^n dk = z \int_0^1 (1 - \ln x)^n\, dx := z I_n. \end{aligned}$$

Integration by parts yields the recurrence relation

$$I_n = 1 + nI_{n-1},$$

with $I_0 = 1$. This solves to $I_n = \lfloor en! \rfloor$ for $n > 0$ [31]. The statement follows. □

This means, algorithm PRODUCT solves any PGLCP instance with a fixed number n of blocks in expected time $O(z) = O(N)$ which is asymptotically optimal.

Algorithm 5 is a close relative of algorithms due to Seidel (for linear programming with n variables and N constraints) [32] and Welzl (for finding the smallest enclosing ball of a set of N points in dimension n) [33].

The expected number of edge evaluations depends on the rule for choosing i, and a bad rule can lead to a complexity which is asymptotically worse than what we found in Theorem 5. For instance, always choosing i to be a lightest dimension yields an expected number of

$$\sum_{k=1}^{z-1} k \left(\prod_{\ell=2}^{k} H_\ell^{a_\ell} \right) \sum_{m=0}^{a_{k+1}-1} H_{k+1}^m \overset{a_z:=n}{=} (z-1) \sum_{m=0}^{n-1} H_z^m = \frac{z-1}{H_z - 1}(H_z^n - 1),$$

edge evaluations, which is superlinear in z. We believe (although we cannot prove it formally) that our choice of i in Algorithm 5 leads to the smallest possible expected number of edge evaluations.

4.2 The Algorithm RandomFacet

The RANDOMFACET algorithm shares its basic idea with the PRODUCT algorithm. In addition to the current set S, it maintains a current vertex $J \in V(S)$ which may be replaced at some point by a neighbor of J along an outgoing edge. This means, we get a *path-following* algorithm. In order to guarantee the invariant $J \in V(S)$, the element j which gets removed from S for the first recursive call must not be in J, see Figure 2 (right).

As in the case of the PRODUCT algorithm, we can derive an explicit bound on the runtime, but here it is an *upper* bound instead of an exact bound. We get that the expected number of edge evaluations for a grid of characteristic $(a_2, \ldots, a_z)$ is at most

$$\sum_{k=1}^{z-1} \prod_{\ell=k}^{z-1} (H_\ell - H_{k-1} + 1)^{a_{\ell+1}} - z + 1.$$

The complexity is again maximized if $a_z = n$, and an upper bound of

$$\sum_{k=1}^{z-1} (H_{z-1} - H_{k-1} + 1)^n - z + 1 = (H_{z-1} + 1)^n + \sum_{k=2}^{z} (H_{z-1} - H_{k-1} + 1)^n - z$$

holds. Comparing this with (12), we see that for large z, both algorithms have approximately the same expected worst-case complexity which we have shown

to be linear in z. However, only RandomFacet has the potential of being faster than the upper bound in practice, for example if the starting vertex is already close to the sink, or if paths tend to be short in the USO under consideration.

The algorithm RandomFacet is a close relative of an algorithm by Matoušek, Sharir and Welzl for *LP-type problems* [20].

5 The Holt-Klee Condition

A grid USO is said to be *Holt-Klee* (HK), if there exists a set of N-n vertex-disjoint paths from source to sink and if in addition, every nonempty subgrid is HK.

In the following, we prove that USO coming from PGLCP are HK. We actually prove that a larger class of digraphs, those defined by *complete pointed fans* in $\mathbb{R}^d$, has the Holt-Klee property. Here, we follow the notation of Ziegler [34].

Definition 7. *A fan in $\mathbb{R}^d$ is a family $\mathcal{F} = \{C_1, C_2, \ldots, C_t\}$ of nonempty polyhedral cones, so that*

(i) Every nonempty face of a cone in $\mathcal{F}$ is also a cone in $\mathcal{F}$.
(ii) The intersection of any two cones in $\mathcal{F}$ is a face of both.

A fan $\mathcal{F}$ is *complete* if the union of its cones is $\mathbb{R}^d$. It is *pointed* if the zero vector is one of its cones. The *dual graph* $G_{\mathcal{F}}$ of a fan has as its vertex set the set of d-dimensional cones of $\mathcal{F}$, with two cones joined by an edge if their intersection is a $(d-1)$-dimensional face of $\mathcal{F}$.

In our PGLCP application (see Section 3), we use $d = N - n$, and we consider a matrix $\hat{A}$ that has as its row space the complement of the row space of A in $\mathbb{R}^N$ and say that a set of columns of $\hat{A}$ generates a cone of $\mathcal{F}$ if and only if the set does not contain all of the columns indexed by Π_i for any i. We show in [1] that property P of the matrix A ensures that the family $\mathcal{F}$ satisfies (i) and (ii). The dual graph of such a fan is a grid graph.

A vector $q \in \mathbb{R}^d$ is said to be in general position with respect to $\mathcal{F}$ if it is not contained in any hyperplane that contains a $(d-1)$-dimensional cone of $\mathcal{F}$. If a vector q is in general position with respect to a fan $\mathcal{F}$, we can define an orientation $\Gamma_{q,\mathcal{F}}$, in which an edge joining cones C and C' is oriented from C to C' if $C \backslash C'$ and q are on opposite sides of the hyperplane containing $C \cap C'$. The digraph $\Gamma_{q,\mathcal{F}}$ has a unique sink and source, which are the faces of $\mathcal{F}$ that contain q and $-q$. The interior of a directed path in a digraph is the set of vertices in the path other than the first and the last vertex.

Theorem 6. *There is a set of d directed paths in $\Gamma_{q,\mathcal{F}}$ from the source to the sink that have pairwise disjoint interiors.*

We need the following lemma for the proof.

Lemma 3. *Let $\mathcal{K} = \{K_1, K_2, \ldots, K_{d-1}\}$ be d-dimensional cones of $\mathcal{F}$, and suppose that none of these cones contains q or $-q$. Then there exists a vector w orthogonal to q so that the segments from q to w and from $-q$ to w both have empty intersection with each of the cones of $\mathcal{K}$.*

Proof. Assume without loss of generality that q is the d^{th} unit vector $(0,\ldots,0,1)$. Let $K \in \mathcal{K}$ and let $\hat{A}_K$ be the submatrix of $\hat{A}$ containing the generators of K. Now let $\overline{A}_K$ be the matrix obtained from $\hat{A}_K$ by deleting the last row. The columns of $\overline{A}_K$ are the projections of the columns of $\hat{A}_K$ onto the hyperplane H_q orthogonal to q. Because neither q nor $-q$ is in K, the systems $\hat{A}_K x = q, x \geq 0$ and $\hat{A}_K x = -q, x \geq 0$ have no solution. It follows that the system $\overline{A}_K x = 0, x \geq 0, x \neq 0$ has no solution. By Gordan's Theorem [35] there exists a vector z_K so that $z_K^T \overline{A}_K > 0$. For such a z_K, any sufficiently small perturbation of it will also satisfy the inequality. Therefore there we can find a linearly independent set $\{z_{K_1}, z_{K_2}, \ldots, z_{K_{d-1}}\}$ so that for each i, $z_{K_i}^T \overline{A}_{K_i} > 0$. Let Z be a $(d-1) \times (d-1)$ matrix that has as its rows the vectors $z_{K_i}^T, i = 1,\ldots,d-1$. Stiemke's Theorem [36] says that the system $Zw \leq 0, Zw \neq 0$ has a solution if and only if the system $y^T Z = 0, y > 0$ has no solution. But the matrix Z is nonsingular, so the second system has no solution. Therefore there must be a vector $w \in H_q$ that is not in any of the cones $\overline{A}_K$ for $K \in \mathcal{K}$. This w is nonzero, because Zw is nonzero. The vector w satisfies the requirements of the lemma, because any intersection of a cone of $\mathcal{K}$ with the segment from q to w or the segment from $-q$ to w would project to the cone generated by w. It should also be noted that since the cones $\overline{A}_K$ are closed, there is an open set of such w. □

With this, we are ready for the proof.

Proof (of Theorem 6). The directed vertex version of Menger's theorem states that there will be d disjoint directed paths from the source of $\Gamma_{q,\mathcal{F}}$ to the sink if and only if there do not exist $d-1$ vertices of the graph other than the source and the sink that cover all directed paths from the source to the sink. A set of $d-1$ vertices of $\Gamma_{q,\mathcal{F}}$ other than the source and the sink corresponds to a set $\mathcal{K} = \{K_1, K_2, \ldots, K_{d-1}\}$ as in the lemma. Because the set of w satisfying the conditions of the lemma is open, we can choose a w for which the segments from w to q and from w to $-q$ do not meet any cones of $\mathcal{F}$ of dimension less than $d-1$. We claim that the sequence of d-dimensional cones met by the directed segment from $-q$ to w, followed by the sequence of d-dimensional cones met by the directed segment from w to q, corresponds to a directed path from the source to the sink of $\Gamma_{q,\mathcal{F}}$. Suppose C_i and C_j are two d-dimensional cones of $\mathcal{F}$, and that the directed segment from $-q$ to w crosses, in order, $(C_i, C_i \cap C_j, C_j)$. Then $C_i \backslash (C_i \cap C_j)$ and q are on opposite sides of the hyperplane spanned by $C_i \cap C_j$, so the edge of $\Gamma_{q,\mathcal{F}}$ connecting C_i and C_j is oriented from C_i to C_j. Similarly, the edges connecting cones met by the directed segment from w to q are oriented consistently with the direction of the segment. □

The grid orientation defined by $\Gamma_{q,\mathcal{F}}$ is the same as that defined in Section 3 through the reduced costs in Equation (3) [1]. This yields the desired result.

Corollary 1. *Any* PGLCP*-induced grid USO ψ satisfies the Holt-Klee condition.*

Reorienting all edges along a fixed dimension of the grid preserves the property of being PGLCP-induced [1]. It follows that a grid USO can only be PGLCP-induced if all 2^n reorientations satisfy the Holt-Klee condition.

The special case of the Theorem 6 in which $\mathcal{F}$ is the normal fan of a polytope was shown by Holt and Klee [24]. Our proof uses Menger's theorem, which was used by Holt and Klee, but does not use the geometry of a polytope. Our approach may be seen as an alternate way to prove the Holt - Klee Theorem. The graph $\Gamma_{q,\mathcal{F}}$ has been used by Kleinschmidt and Onn to prove that fans are *signable* [19]. Restricted to PGLCP-induced fans, their result simply says that the undirected grid graph Γ_F underlying $\Gamma_{q,\mathcal{F}}$ has a unique sink orientation, namely $\Gamma_{q,\mathcal{F}}$. Theorem 6 strengthens the result of Kleinschmidt and Onn by showing that the signings they produce have an interesting additional property.

Acknowledgment

We thank an IPCO referee for useful comments, Falk Tschirschnitz for many helpful discussions and David Loeffler for suggesting the simple proof of Lemma 2.

References

1. Gärtner, B., Morris, Jr., W.D., Rüst, L.: Unique sink orientations of grids. Technical Report 472, Institute of Theoretical Computer Science (2005)
2. Cottle, R.W., Dantzig, G.B.: A generalization of the linear complementarity problem. Journal on Combinatorial Theory **8** (1970) 79–90
3. Cottle, R.W., Pang, J., Stone, R.E.: The Linear Complementarity Problem. Academic Press (1992)
4. Megiddo, N.: A note on the complexity of P-matrix LCP and computing an equilibrium. Technical report, IBM Almaden Research Center, San Jose (1988)
5. Khachiyan, L.G.: Polynomial algorithms in linear programming. U.S.S.R. Comput. Math. and Math. Phys **20** (1980) 53–72
6. Adler, I., Saigal, R.: Long monotone paths in abstract polytopes. Math. Operations Research **1** (1976) 89–95
7. Kalai, G.: Linear programming, the simplex algorithm and simple polytopes. Math. Programming **79** (1997) 217–233
8. Wiedemann, D.: Unimodal set-functions. Congressus Numerantium **50** (1985) 165–169
9. Hoke, K.W.: Completely unimodal numberings of a simple polytope. Discrete Appl. Math. **20** (1988) 69–81
10. Kalai, G., Kleitman, D.J.: A quasi-polynomial bound for the diameter of graphs of polyhedra. Bulletin Amer. Math. Soc. **26** (1992) 315–316
11. Kalai, G.: A simple way to tell a simple polytope from its graph. J. Combinatorial Theory **49** (1988) 381–383
12. Björklund, H., Sandberg, S., Vorobyov, S.: Randomized subexponential algorithms for parity games. Technical Report TR-2003-019, Department of Information Technology (2003)
13. Björklund, H., Sandberg, S., Vorobyov, S.: A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. In: Proc. 29th International Symposium on Mathematical Foundations of Computer Science (MFCS). Volume 3153 of Lecture Notes in Computer Science., Springer-Verlag (2004) 673–685

14. Grädel, E., Thomas, W., Wilke, T.: Automata, Logics, and Infinite Games. Volume 2500 of Lecture Notes in Computer Science. Springer-Verlag (2002)
15. Tschirschnitz, F.: LP-related Properties of Polytopes with Few Facets. PhD thesis, ETH Zürich (2003)
16. Felsner, S., Gärtner, B., Tschirschnitz, F.: Grid orientations, $(d, d+2)$-polytopes, and arrangements of pseudolines. Discrete and Computational Geometry (to appear)
17. Stickney, A., Watson, L.: Digraph models of Bard-type algorithms for the linear complementarity problem. Mathematics of Operations Research **3** (1978) 322–333
18. Szabó, T., Welzl, E.: Unique sink orientations of cubes. In: Proc. 42^{nd} IEEE Symp. on Foundations of Comput. Sci. (2000) 547–555
19. Kleinschmidt, P., Onn, S.: Signable posets and partitionable simplicial complexes. Discrete Comput. Geom. **15** (1996) 443–466
20. Matoušek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. Algorithmica **16** (1996) 498–516
21. Chazelle, B., Matoušek, J.: On linear-time deterministic algorithms for optimization problems in fixed dimension. J. Algorithms **21** (1996) 579–597
22. Clarkson, K.L.: Las Vegas algorithms for linear and integer programming. J. ACM **42** (1995) 488–499
23. Gärtner, B., Welzl, E.: Linear programming – randomization and abstract frameworks. In: Proc. 13th Ann. ACM Symp. Theoretical Aspects of Computer Science. Volume 1046 of Lecture Notes Comput. Sci., Springer-Verlag (1996) 669–687
24. Holt, F., Klee, V.: A proof of the strict monotone 4-step conjecture. In Chazelle, J., Goodman, J.B., Pollack, R., eds.: Advances in Discrete and Computational geometry. Contemporary Mathematics. Amer. Math. Soc. (1998)
25. Matoušek, J.: Lower bounds for a subexponential optimization algorithm. Random Structures & Algorithms **5** (1994) 591–607
26. Gärtner, B.: The Random-Facet simplex algorithm on combinatorial cubes. Random Structures & Algorithms **20** (2002)
27. Schäfer, U.: A linear complementarity problem with a P-matrix. SIAM Review **46** (2004) 189–201
28. Chvátal, V.: Linear Programming. W. H. Freeman, New York, NY (1983)
29. Szanc, B.P.: The Generalized Complementarity Problem. PhD thesis, Rensselaer Polytechnic Institute, Troy, NY (1989)
30. Gowda, M.S., Sznajder, R.: The generalized order linear complementarity problem. SIAM J. Matrix Anal. Appl. **15** (1994) 779–795
31. Sloane, N.J.A.: Sequence A000522. The On-Line Encyclopedia of Integer Sequences, `http://www.research.att.com/~njas/sequences/` (2004)
32. Seidel, R.: Small-dimensional linear programming and convex hulls made easy. Discrete Comput. Geom. **6** (1991) 423–434
33. Welzl, E.: Smallest enclosing disks (balls and ellipsoids). In Maurer, H., ed.: New Results and New Trends in Computer Science. Volume 555 of Lecture Notes Comput. Sci. Springer-Verlag (1991) 359–370
34. Ziegler, G.M.: Lectures on Polytopes. Springer-Verlag (1995)
35. Gordan, P.: Über die Auflösung linearer Gleichungen mit reellen Coefficienten. Mathematische Annalen **6** (1873) 23–28
36. Stiemke, E.: Über positive Lösungen homogener linearer Gleichungen. Journal für die reine und angewandte Mathematik **76** (1915) 340–342

Jumping Doesn't Help in Abstract Cubes

Ingo Schurr* and Tibor Szabó**

Theoretical Computer Science, ETH Zürich,
CH-8092 Zürich, Switzerland
{schurr, szabo}@inf.ethz.ch

Abstract. We construct a class of abstract objective functions on the cube, such that the algorithm BottomAntipodal takes exponentially many steps to find the maximum. A similar class of abstract objective functions is constructed for the process BottomTop, also requiring exponentially many steps.

1 Introduction

The Model. Let $P \subseteq \mathbb{R}^n$ be a convex polyhedron (given as the intersection of m halfspaces) and $c : \mathbb{R}^n \to \mathbb{R}$ be a linear objective function; *linear programming* seeks a vertex of P maximizing c. While linear programming is known to have a polynomial time algorithm in the *bit size* of the input, its complexity in the so-called *unit-cost* model remains an important open question. That is, what is the smallest $f(n, m)$ such that any linear program in dimension n with m constraints can be solved in time at most $f(n, m)$ if all arithmetic operations are assumed to incur unit cost?

Numerous researchers studied this problem and still our understanding is far from satisfactory. The best known algorithms, due to Kalai [9] and Matoušek, Sharir, and Welzl [11], work in time $e^{O(\sqrt{m \log n})}$. An important aspect of both approaches is that they disregard most of the geometric content of the problem and consider only a basic combinatorial skeleton.

One of the most natural and useful combinatorial simplifications is the concept of *abstract objective functions*, which was first introduced by Adler and his coauthors [1, 2], and later by Williamson Hoke [15] and Kalai [8] under different names. Given a convex polytope P with vertex set V, the *graph* of P is the graph $G(P)$ with vertex set V and with edges corresponding to the edges (1-dimensional faces) of P. A function $f : V \to \mathbb{R}$ is called an abstract objective function if on every face F of P there is a unique local maximum of f. That is, there is a unique vertex $v \in F$, such that v has a larger f-value than all its neighbors in $G(P)$. In particular, this unique local maximum is a global maximum on F.

* Supported by the joint Berlin/Zürich graduate program Combinatorics, Geometry, and Computation (CGC), financed by German Science Foundation (DFG) and ETH Zurich.

** Research is supported by the SNF grant 200021-100316/1.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 225–235, 2005.

Note that a generic linear function is a special case of an abstract objective function, though most abstract objective functions of a given polytope cannot be achieved by a linear function. The model is still quite powerful: Kalai's subexponential randomized simplex algorithm for linear programming works for arbitrary abstract objective functions.

In our paper we prefer to formulate our results in an alternative but equivalent framework. An orientation of the edges of the graph $G(P)$ is called an *acyclic unique sink orientation*[1] or *AUSO* of the polytope P if the induced subgraph of $G(P)$ on the vertex set of any face of P has exactly one sink (vertex of outdegree zero). We assume that the orientation is given by an oracle: after querying a vertex the oracle returns the orientation of the edges incident to the queried vertex.

Any abstract objective function induces an AUSO: Orient every edge from the vertex with the smaller value to the one with the larger value. The optimum vertex with the largest value of the objective function becomes the (unique) sink of $G(P)$. On the other hand, given an AUSO, the vertices of $G(P)$ can be numbered according to an arbitrary linear order which extends the partial order defined by the AUSO. This numbering defines an abstract objective function on P.

The Algorithm. The oldest and most natural combinatorial attack to solve a linear program is the *simplex algorithm*. Geometrically, it can be viewed as follows: We start at some initial vertex of the polytope P and at each step we move from the current vertex v along an edge of P to another vertex w with $c(w) > c(v)$ (this is called a *pivot step*). Typically there are several possible choices of w at each step, and the way of selecting one of them is called a *pivot rule*. The simplex algorithm terminates for every pivot rule, of course, but the difference in the number of steps for different pivot rules may be enormous.

Earlier results on the worst-case complexity of various pivot rules are rather discouraging. For Dantzig's original pivot rule, Klee and Minty [10] constructed a class of examples where this rule leads to an exponential number of steps. It is a polytope isomorphic to the cube $[0, 1]^n$, but the cube is slightly deformed in such a way that there is a Hamiltonian monotone path, that is, a directed path visiting all vertices such that a suitable linear objective function increases along it. Subsequently such worst-case examples were found by various researchers for almost all known deterministic pivot rules; see Goldfarb [6] for an overview and Amenta and Ziegler [3] for a new unified view of these examples.

Motivated by the unsuccessful attempts of deterministic simplex algorithms, Kaibel [7] suggested a very non-simplex-like algorithm, something which takes not only one of the outgoing edges into account, but, in some sense all of them. Of course in order to abandon the idea of progress along an edge, one needs to know something about the structure of the polytope. For example, when our polytope is combinatorially equivalent to the n-dimensional cube, then it makes sense to

[1] For some purposes, it is also very interesting to consider unique-sink orientations of polytopes that are not necessarily acyclic (see, e.g., [12, 14, 13]).

speak about an "antipodal vertex". Each vertex is at the bottom of the face generated by its outgoing edges, as it is the source of this face. Motivated by the picture of the orthogonal cube, the algorithm BottomAntipodal jumps from the bottom vertex of the face generated by its outgoing edges to the antipodal vertex within this face.

We also consider a process called BottomTop (also suggested by Kaibel [7]), which, in some sense, represents the most greedy approach one can imagine. Being at a vertex v and knowing the adjacent outgoing edges, one knows that every vertex in the face generated by these outgoing edges is *better* than the current vertex. Suppose we have access to an oracle which tells us the *best* vertex, i.e., the sink, in this subcube, and thus we are able to jump there in one step. It is then plausible to believe this to be a good idea. A step of the process BottomTop is defined by jumping from the current vertex v to the sink v' in the subcube generated by the outgoing edges incident to v. BottomTop is of course *not* an algorithm, since we need to have access to an oracle which tells us the sink of a subcube once we provide the source.

The Results. In Sect. 4 of this paper we construct an acyclic unique sink orientation of the n-dimensional cube, such that BottomAntipodal takes an exponential number of queries to find the sink. In Sect. 5 we give a construction of an AUSO of the cube on which BottomTop performs an exponential number of queries.

2 Preliminaries on AUSOs

Let $e_i \in \{0,1\}^n$ be the vector having 1 at position i and zeros elsewhere. For zero-one vectors v and w, $v+w$ is understood as the modulo 2 sum of v and w. The notation vw stands for the concatenation of the vectors v and w. The zero vector of dimension greater than one is denoted by $\mathbf{0}$ and the reader is trusted to figure out the correct length of the vector.

From now on, by an AUSO we will mean an acyclic unique-sink orientation of the cube $[0,1]^n$ (we will not consider any other polytopes). The graph of the n-dimensional cube is the usual n-dimensional (graph-theoretic) cube with vertex set $\{0,1\}^n$. The neighbors of a vertex v are $v+e_i$, $i=1,2,\ldots,n$.

Formally we will identify an n-dimensional AUSO A with its *outmap* $s_A : \{0,1\}^n \to \{0,1\}^n$, where $s_A(v)_i = 1$ if the edge $\{v, v+e_i\}$ is oriented from v towards $v+e_i$, and $s_A(v)_i = 0$ otherwise, i.e., if that edge is oriented from $v+e_i$ towards v. It is known that the outmap s_A is a bijection for any AUSO A, even if we restrict it to an arbitrary subcube. In particular, an AUSO does not only have a unique sink per face, but also, e.g., a unique source. For this and other facts about unique sink orientations of cubes see, for example, [14].

We say that two AUSOs A and B are *isomorphic* if there is a bijection between the vertices of A and the vertices of B that preserves the oriented edges.

In the following we describe two lemmas, special cases of results of [13], which allow us to construct new AUSOs from old ones. The first lemma uses the product structure of the cube.

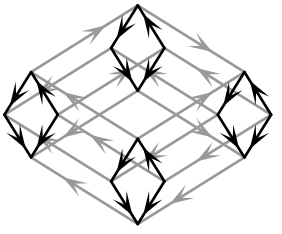


Fig. 1. The blowup-construction: Four identical 2-dimensional AUSO (drawn in black) are interconnected by 2^2 2-dimensional frames (drawn in gray)

Lemma 1 (Blowup construction, [13–Lemma 3]). *Let A be an AUSO of dimension m and for each $u \in \{0,1\}^m$ let B_u be a n-dimensional AUSO. Then the map $s_C : \{0,1\}^{m+n} \to \{0,1\}^{m+n}$ defined by $s_C(uv) = s_A(u)s_{B_u}(v)$ is the outmap of an $(m+n)$-dimensional AUSO C.*

One can imagine that we blow up each vertex of A to a n-dimensional cube, which is oriented according to some AUSO, generally different for different vertices. For us, however, a complementary view will be more useful: We can obtain C by taking 2^n copies of A and, for each vertex u of A, interconnecting all the 2^n copies of u by an n-dimensional cubic "frame" oriented according to B_u. This is illustrated in Fig. 1.

The second lemma, the heart of our recursion, allows to change the orientation on a smaller subcube under appropriate conditions. Let A be an n-dimensional AUSO and let S be a face of the n-dimensional cube (isomorphic to an m-dimensional cube for some $m \leq n$). We call S a *hypersink* of A if all edges connecting vertices of S to vertices outside S are oriented towards S.

Lemma 2 (Hypersink reorientation, [13–Lemma 5]). *Let A be an n-dimensional AUSO and let S be an m-dimensional hypersink of A. If the edges within S are reoriented according to an arbitrary m-dimensional AUSO B, and the orientations of all other edges are left as in A, then the resulting orientation of the n-dimensional cube is an AUSO.*

As a warm-up let us recall the definition of the Klee-Minty cube in the framework of our lemmas. The zero-dimensional Klee-Minty cube KM_0 consists of one vertex. To construct KM_n we take two copies K and K' of KM_{n-1} and flip the orientations of all edges in one of them, say in K'. Then we add a perfect matching between the vertices of K and K' having identical coordinates and orient these edges from K' towards K . Note that by Lemma 1 the resulting orientation is an AUSO: We interconnect 2^{n-1} copies of a 1-dimensional USO using the two frames K and K'. Originally the Klee-Minty cube was defined as a geometric object. It is a polytope combinatorially equivalent to the cube, which produces the orientation defined above if we evaluate an appropriate linear objective function on its vertices. (See, e.g, [4] for a more detailed study on Klee-Minty cubes.) As we mentioned in the introduction the Klee-Minty cube is a worst-case example for many of the natural pivot rules of the simplex algorithm. As it turns out, BottomAntipodal and BottomTop are very efficient on the Klee-Minty cube, even on a much wider class of AUSOs called *decomposable* orientations (See [15] for an introduction to decomposable AUSOs).

Lemma 3. *Starting at an arbitrary vertex,* BottomTop *needs at most* $n+1$ *steps to find the sink of the* n*-dimensional Klee-Minty cube.*

Due to the construction of the Klee-Minty cubes for the coordinate n all edges are oriented towards the same facet S_0. In particular *all* subcubes have their sink in S_0, so after the first step BottomTop will be in S_0. Repeating this argument inductively, after $n+1$ steps we are in the sink. A similar argument works for BottomAntipodal (see [13–Proposition 7]).

3 The BottomAntipodal Tree

The behavior of BottomAntipodal on an AUSO A can be described by the following directed graph $T_{\mathrm{ba}}(A)$: The vertex set of $T_{\mathrm{ba}}(A)$ is the vertex set of the underlying cube of A. Two vertices v, w form an edge $v \to w$ if $v + s_A(v) = w$. In particular, every vertex v has exactly one outgoing edge $v \to v + s_A(v)$. We call $v + s_A(v)$ the *successor* of v and denote it by $\mathrm{succ}_A(v)$. The sink o of s_A is special since it is the only vertex in $T_{\mathrm{ba}}(A)$ having a loop. For an example, see Fig. 2.

The unique path starting in a vertex v in $T_{\mathrm{ba}}(A)$ will be called the *trace* of v. Obviously, the trace of a vertex v is the sequence of queries BottomAntipodal produces starting in v.

BottomAntipodal will terminate on any AUSO, i.e., $T_{\mathrm{ba}}(A)$ is a tree. The vertex v is a source in the cube spanned by v and $v+s_A(v)$. The following lemma provides us with a path from v to $v+s_A(v)$ in the AUSO A. (The proof is rather easy and omitted.)

Lemma 4. *There is a path from the source of an AUSO to any vertex of the cube.*

By the above lemma, the trace of a vertex in $T_{\mathrm{ba}}(A)$ induces a trail in A. In particular, if $T_{\mathrm{ba}}(A)$ contains a cycle, so does A. Hence for an AUSO A the graph $T_{\mathrm{ba}}(A)$ is a tree, the so-called *bottom-antipodal tree*. This tree was first introduced by Kaibel [7] in connection to randomized simplex algorithms.

Since $T_{\mathrm{ba}}(A)$ is a tree, the trace of a vertex v is a path in $T_{\mathrm{ba}}(A)$ to the sink of A. The length of this path is the *height* $h_A(v)$ of v. Obviously, $h_A(v)$ differs

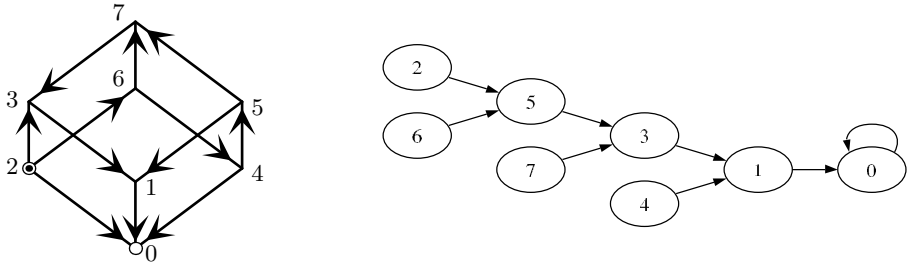


Fig. 2. An AUSO A with a graph $T_{\mathrm{ba}}(A)$ of height 4

by one from the number of queries of BottomAntipodal starting in v. The height $h(A)$ of $T_{\mathrm{ba}}(A)$ is defined as the maximal height of a vertex in A and the average height $\tilde{h}(A)$ is the average over the heights of all vertices in A, that is, for an n-dimensional A we have

$$h(A) = \max\{h_A(v) \mid v \in \{0,1\}^n\} \qquad \tilde{h}(A) = \frac{1}{2^n} \sum_{v \in \{0,1\}^n} h_A(v).$$

The height of A corresponds to the worst-case behavior of BottomAntipodal, whereas the average height reflects the expected behavior if we randomize the starting vertex. Thus, if we can construct a family of examples for which the maximal height grows exponentially, then BottomAntipodal has exponential worst-case complexity.

4 The Construction

Theorem 1. *Let A be an arbitrary n-dimensional AUSO. Then there exists a $(n+2)$-dimensional AUSO D, such that the height of D is at least $2h(A)$ and the average height of D is at least $\frac{3}{4}\tilde{h}(A) + \frac{1}{2}h(A)$.*

Starting, e.g., with $A_2 = KM_2$ and the orientation A_3 from Fig. 2, Theorem 1 yields a sequence of AUSOs with exponential height.

Corollary 1. *In every dimension $n \geq 2$ there is an AUSO, such that the height of the corresponding bottom-antipodal tree is at least $\sqrt{2}^n$ and the average height is at least $\frac{2}{5}\sqrt{2}^n$.*

Proof (of Theorem 1).

We can assume without loss of generality that the sink of A is in 0. As an intermediate step we first construct an $(n+2)$-dimensional AUSO C, which is the blow-up of A. For each vertex $u \in \{0,1\}^n$ we select a 2-dimensional AUSO B_u which is isomorphic to KM_2. We do not, however, select identical copies. Let $K_0 = KM_2$ be the 2-dimensional Klee-Minty cube, while K_1 be KM_2 with its two coordinates permuted. (Fig. 3)

Now let $B_u = K_0$ if the height $h_A(u)$ is even and let $B_u = K_1$ if $h_A(u)$ is odd. We let C be the blowup of A by these B_u. So, according to our preferred view of the blowup construction, we take 4 copies of A and interconnect them by the 2-dimensional frames B_u, each is the Klee–Minty cube with possibly permuted coordinates. All B_u have their sink at $u00$, that is, in the same n-dimensional face S_0, which is a copy of A. Consequently, this copy of A is a hypersink in C.

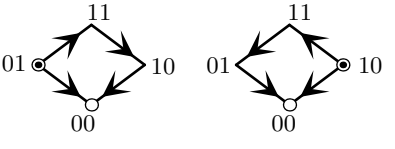


Fig. 3. The two orientations K_0 and K_1

Table 1. The outmap of the constructed AUSO D in the last two coordinates and the color-coding of the vertices of D

$''$	00	11		10		01	
$h_A(\ ')$		even	odd	even	odd	even	odd
$_D(\ )''$	00	01	10	10	11	11	01
color	blue	green		red	yellow	yellow	red

Let v_{max} be a vertex of A with $h_A(v_{max}) = h(A)$. We now reorient the hypersink S_0 by rotating v_{max} to 0 (which used to be the sink in S_0). More formally, we obtain a new orientation A' isomorphic to A, where the outmap of a vertex z is obtained by $s_{A'}(z) = s_A(z+w)$. Then we orient the hypersink S_0 of C according to A' and we denote the resulting $(n+2)$-dimensional AUSO by D.

We can also express the outmap of D formally. For an $(n+2)$-dimensional vector v denote by v' the projection of v to the first n coordinate, while v'' is the projection of v to the last 2 coordinates. Then $s_D(v)' = s_A(v')$ unless $v'' = 0$ in which case $s_D(v)' = s_A(v' + v_{max})$. The last two coordinates $s_D(v)''$ of the outmap $s_D(v)$ only depend on the last two coordinates of v and the parity of $h_A(v')$. See Table 1 for the details.

We introduce a color-coding of the vertices depending on where they lie in the 2-dimensional Klee-Minty frame they belong to. The hypersink S_0, having vertices with last two coordinates 00, is composed of blue vertices only. The green vertices occupy the n-dimensional subcube S_3 with last two coordinates 11. The remaining two subcubes, S_1 and S_2 are occupied by yellow and red vertices, depending on the parity of the height of the projection of the particular vertex in A. Crucially, $v \in S_1 \cup S_2$ and $v + \mathbf{0}11$, being in the same Klee-Minty frame, have different colors; one of them is red, the other is yellow. See Table 1 on how the vertices are colored. Fig. 4 aims to illustrate the construction and color-coding.

In the following we make a few observations about the successor $\mathrm{succ}_D(v)$ of a vertex $v \in \{0,1\}^{n+2}$ in the bottom antipodal tree of D.

The successor of a blue vertex is blue. Moreover if v is blue then $\mathrm{succ}_D(v)' = \mathrm{succ}_{A'}(v')$. In other words once BottomAntipodal is in the hypersink S_0, it never leaves S_0 and follows the trace of A' (which is isomorphic to A). In particular $h_D(v) = h_{A'}(v') = h_A(v + v_{max})$.

Next we check the height of the four vertices with $v' = \mathbf{0}$. By construction, the vertices $\mathbf{0}01$, $\mathbf{0}10$, and $\mathbf{0}11$ are the sinks of the copies of A in S_1, S_2, S_3, respectively. In the subcube S_0 we rotated A, such that the sink and $v_{\max}$ change position. Hence, $\mathbf{0}00$ has height $h_D(\mathbf{0}00) = h_A(v_{max}) = h(A)$. The vertex $\mathbf{0}10$ has only its $(n+1)$-edge outgoing. Thus, its successor is $\mathbf{0}00$ and $h_D(\mathbf{0}10) = 1 + h_D(\mathbf{0}00) = 1 + h(A)$. The vertex $\mathbf{0}01$ has outgoing edges along coordinate $(n+1)$ and $(n+2)$ and its successor is $\mathbf{0}10$. Therefore $h_D(\mathbf{0}01) = 1 + h_D(\mathbf{0}10) = 2 + h(A)$. Finally, the vertex $\mathbf{0}11$ has only its $(n+2)$-edge outgoing and $h_D(\mathbf{0}11) = 1 + h_D(\mathbf{0}10) = 2 + h(A)$.

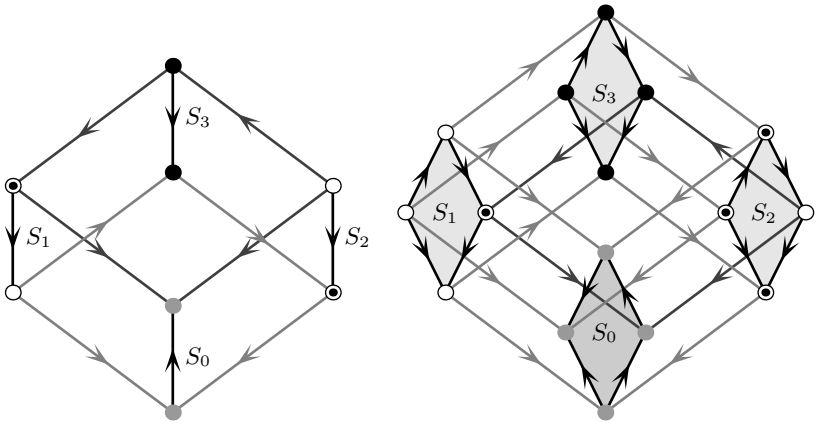


Fig. 4. The orientations A_3 and A_4. The orientation A_3 to the left is constructed from the 1-dimensional AUSO A_1 with sink in $\mathbf{0}$. The orientation A_4 to the right is based on A_2. The subcubes S_1, S_2, and S_3 contain A_1 and A_2, respectively, whereas S_0 contains a flipped variant. The colors are encoded the following way: yellow ○, green ●, blue ●, red ◉

The remaining vertices are considered by their color. Assume now that v is not blue and $v' \neq \mathbf{0}$.

If v is red then its successor is blue.

If v is yellow then its successor is also yellow. Moreover $\mathrm{succ}_D(v)' = \mathrm{succ}_A(v')$. A yellow vertex is either in S_1 or S_2. Since $s_D(v)'' = 11$, the successor of v is in the antipodal n-face. Moreover $\mathrm{succ}_D(v)' = \mathrm{succ}_A(v')$, since the copies of A in S_1 and S_2 are translates of each other. Now $\mathrm{succ}_D(v)$ is yellow because the height of $\mathrm{succ}_D(v)'$ is one less, thus has a different parity than the height of v'. Thus, the trace of v reaches the sink of S_2 through yellow vertices while the first n coordinates are going through the trace of v' in A. Hence, we have that $h_D(v) = h_A(v') + 1 + h_D(\mathbf{0}10) = h_A(v') + h(A) + 2$.

If v is green then its successor is yellow. Moreover $\mathrm{succ}_D(v)' = \mathrm{succ}_A(v')$.

The argument is similar to the above. As v is in S_3 and $s_D(v)'' = 01$ or 10, the successor of v is either in S_1 or S_2 Moreover $\mathrm{succ}_D(v)' = \mathrm{succ}_A(v')$, since the copies of A in S_3, S_1, and S_2 are translates of each other. Now $\mathrm{succ}_D(v)$ is yellow: The height of $\mathrm{succ}_D(v)'$ has a different parity than the height of v'. Hence, $\mathrm{succ}_D(v)$ is in a different frame than v and $s_D(\mathrm{succ}_D(v))'' = 11$. In fact the successor of v is identical to the successor of the yellow vertex w in the 2-dimensional Klee-Minty frame v belongs to. Quantitatively, $h_D(v) = h_D(w) = h_A(w') + h(A) + 2 = h_A(v') + h(A) + 2$.

Now consider the green vertex u with $u' = v_{\max}$. By the above we obtain that

$$h(D) \geq h_D(u) = h_A(u') + h(A) + 2 = 2h(A) + 2.$$

For the average height $\tilde{h}(D)$ we forget about the red vertices (as we don't know their height) and get (with B being the set of blue vertices, Y the set of yellow vertices and G the set of green vertices)

$$\begin{aligned}\tilde{h}(D) &= \frac{1}{2^{n+2}}\sum_v h_D(v) \geq \frac{1}{2^{n+2}}\left(\sum_{v\in B} h_D(v) + \sum_{v\in Y} h_D(v) + \sum_{v\in G} h_D(v)\right)\\ &= \frac{1}{4}\cdot\frac{1}{2^n}\sum_{v\in B} h_{A'}(v') + \frac{1}{4}\cdot\frac{1}{2^n}\sum_{v\in Y} h_A(v') + \frac{1}{4}\cdot\frac{1}{2^n}\sum_{v\in G} h_A(v') + \frac{h(A)}{2} + 1\\ &= \frac{3}{4}\tilde{h}(A) + \frac{1}{2}h(A) + 1.\end{aligned}$$

□

5 The BottomTop Tree

Recall that the process BottomTop in each step jumps to the sink of the subcube spanned by the outgoing edges of the current vertex. We define a directed graph $T_{\mathrm{bt}}(A)$, similar to $T_{\mathrm{ba}}(A)$. The vertex set of $T_{\mathrm{bt}}(A)$ is the vertex set of A. Two vertices u and v are connected by an edge $u \to v$, if v is the sink in the subcube spanned by u and the outgoing edges incident to u. Again, we call v the successor of u. Since A has a unique sink in every subcube, each vertex u has exactly one successor v. Also, since A is acyclic, by Lemma 4 again $T_{\mathrm{bt}}(A)$ is a tree, the *bottom top tree*. Let $t_A(v)$ be the height of v in $T_{\mathrm{bt}}(A)$, $t(A)$ be the height of $T_{\mathrm{bt}}(A)$ and $\tilde{t}(A)$ be the average height of $T_{\mathrm{bt}}(A)$.

Theorem 2. *Let A be an arbitrary n-dimensional AUSO. Then there exists a $(n+2)$-dimensional AUSO D, such that the height of the bottom top tree of D is at least $2t(A)$ and the average height is at least $\frac{1}{2}\tilde{t}(A) + \frac{1}{4}t(A)$.*

Proof. We repeat the construction in Theorem 1, but now we use t instead of h. That is, v_{max} now is of maximal height with respect to t_A and the copies of K_0 and K_1 for the frame are chosen according to the parity of $t_A(u)$.

With the same color coding as in Theorem 1 we now have the following picture. As before, blue vertices have blue successors and $\mathrm{succ}_D(v)' = \mathrm{succ}_{A'}(v')$ provided v is a blue vertex. The vertex $\mathbf{0}00$ corresponds to v_{max} in A, hence $t_D(\mathbf{0}00) = t(A)$. The vertices $\mathbf{0}01$ and $\mathbf{0}10$ have successor $\mathbf{0}00$ and thus $t_D(\mathbf{0}01) = 1 + t(A) = t_D(\mathbf{0}10)$. Finally, $\mathbf{0}11$ has successor $\mathbf{0}10$.

Assume that $v' \neq \mathbf{0}$. Red and yellow vertices all have blue successors. The major difference, compared to the proof of Theorem 1 is that now the green vertices have green successors. A green vertex v has exactly one of the edges along coordinates $n+1$ and $n+2$, say $n+1$, outgoing. Thus, the subcube B spanned by v and its outgoing edges consists of two facets, each is a translate of the subcube B', spanned by v' and its outgoing edges in A. The sink of B is either in the vertex $w'01$ or $w'11$, where w' is the sink of B'. Since $w' = \mathrm{succ}_A(v')$, the value $t_A(w')$ is one less, thus has a different parity than $t_A(v')$. Thus the edge

between $w'01$ and $w'11$ is oriented towards $w'11$, which is then the sink of B and the successor of v. Note, that $w'11$ is a green vertex and $\text{succ}_D(v)' = \text{succ}_A(v')$.

For $u = v_{max}11$ we now obtain $t_D(u) = t_A(v_{max}) + 2 + t(A) = 2t(A) + 2$.

The average height of D we estimate by forgetting about all the red and yellow vertices. Summing over the green and blue vertices we get an estimate of

$$\tilde{t}(D) \geq \frac{1}{2}\tilde{t}(A) + \frac{1}{4}t(A). \qquad \square$$

Easy calculation shows that $t(A_2) = 2$, $\tilde{t}(A_2) = 1$, $t(A_3) = 4$, and $\tilde{t}(A_3) = 7/4$. This yields the following corollary.

Corollary 2. *In every dimension $n \geq 2$ there is an AUSO, such that the height of the corresponding* BottomTop *tree is at least $\sqrt{2}^n$ and the average height is at least $\frac{1}{6}\sqrt{2}^n$.*

6 Remarks and Open Problems

1.In [14] unique sink orientations of cubes were investigated, which are not necessarily acyclic. As it turns out this more general model is quite useful, for example it contains linear programming in its whole generality, not just on polytopes combinatorially equivalent to the cube. This connection is more abstract than the obvious relation to AUSOs, more details about it are found in [5]. BottomAntipodal is also possible to perform on this more general model, except that it is even less useful than for AUSOs since there BottomAntipodal can cycle.
2. It is an intriguing open question to decide what happens when BottomAntipodal or BottomTop is performed on a realizable AUSO, that is on a polytope combinatorially equivalent to the cube. Note that for the Klee-Minty-cube, and in fact for the much larger class of decomposable orientations, both processes have a worst case running time of $n + 1$.

Also, we don't know whether it is possible to extend our construction to AUSOs which satisfy the Holt-Klee condition. (The Holt-Klee condition, satisfied by all realizable polytopes, requires that there are d edge-disjoint source-to-sink paths in any d-dimensional face of the polytope.)

Acknowledgment

We would like to thank Emo Welzl for stimulating discussions.

References

1. I. Adler, G. B. Dantzig. Maximum diameter of abstract polytopes. *Math. Programming Studies*, 1:20–40, 1974.
2. L. Adler, R. Saigal, Long monotone paths in abstract polytopes *Math. Oper. Res.*, 1:89–95, 1976.

3. N. Amenta and G. M. Ziegler. Shadows and slices of polytopes. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 10–19, 1996.
4. B. Gärtner, M. Henk, and G. M. Ziegler. Randomized simplex algorithms on Klee-Minty cubes. *Combinatorica*, 18(3):349–372, 1998.
5. B. Gärtner and I. Schurr. Linear Programming and Geometric Unique Sink Orientations. Manuscript in preparation, 2004.
6. D. Goldfarb. On the complexity of the simplex algorithm. In *Advances in optimization and numerical analysis*, pages 25–38, Dordrecht, 1994. Kluwer.
7. V. Kaibel. *The La Claustra lecture.*
8. G. Kalai. A simple way to tell a simple polytope from its graph. *J. Combin. Theory, Ser. A*, 49(2):381–383, 1988.
9. G. Kalai. A subexponential randomized simplex algorithm. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 475–482, 1992.
10. V. Klee, G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities III*, pages 159–175. Academic Press, 1972.
11. J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.
12. W. D. Morris. Randomized principal pivot algorithms for P-matrix linear complementarity problems. *Mathematical Programming*, Ser. A, 92:285–296, 2002.
13. I. Schurr, T. Szabó. Finding the sink takes some time, *Discrete and Computational Geometry*, to appear. An extended abstract appeared in the proceedings of the *European Symposium on Algorithms (ESA)*, 2002.
14. T. Szabó, E. Welzl. Unique sink orientations of cubes. In *Proc.* 42^{nd} *IEEE Symp. on Foundations of Comput. Sci.*, pages 547–555, 2001.
15. K. Williamson Hoke. Completely unimodal numberings of a simple polytope. *Discrete Appl. Math.*, 20:69–81, 1988.

Computing the Inertia from Sign Patterns

Naonori Kakimura and Satoru Iwata

Department of Mathematical Informatics,
Graduate School of Information Science and Technology,
University of Tokyo, Tokyo 113-8656, Japan
{naonori_kakimura, iwata}@mist.i.u-tokyo.ac.jp

Abstract. A symmetric matrix A is said to be sign-nonsingular if every symmetric matrix with the same sign pattern as A is nonsingular. Hall, Li and Wang showed that the inertia of a sign-nonsingular symmetric matrix is determined uniquely by its sign pattern. The purpose of this paper is to present an efficient algorithm for computing the inertia of such matrices. The algorithm runs in O( ) time for a symmetric matrix of order with nonzero entries. The correctness of the algorithm provides an alternative proof of the result by Hall *et al.* In addition, for a symmetric matrix in general, it is shown to be NP-complete to decide whether the inertia of the matrix is not determined by the sign pattern.

1 Introduction

The inertia of a symmetric matrix indicates the number of the positive/negative eigenvalues. This is an important quantity invariant under the congruence transformation. Quadratic forms are classified by the inertia of the coefficient matrices. In this paper, we discuss computing the inertia of a given symmetric matrix from the sign pattern of its entries without numerical information.

Matrix analysis by sign patterns has been studied by many researchers (see [2]). The analysis is called *qualitative matrix theory*. It is useful because the sign pattern of a matrix can be easily inferred in a variety of situations, rather than the exact value of the matrix entries. For a matrix A, we denote by $\mathcal{Q}(A)$ the set of all matrices having the same sign pattern as A, which is called the *sign pattern class* of A. Klee, Ladner and Manber [5] proved that for a rectangular matrix A, it is NP-complete to discern whether there exists $\tilde{A} \in \mathcal{Q}(A)$ which is not row full-rank. However, Robertson, Seymour and Thomas [9] devised a polynomial time algorithm to discern whether $\tilde{A}$ is nonsingular for any square matrix $\tilde{A} \in \mathcal{Q}(A)$.

This paper deals with the sign pattern class of symmetric matrices, which is denoted by $\mathcal{Q}^*(A) := \{\tilde{A} \mid \tilde{A} \in \mathcal{Q}(A), \tilde{A}^\top = \tilde{A}\}$. A symmetric matrix A is said to be *sign-nonsingular* if $\tilde{A}$ is nonsingular for any symmetric matrix $\tilde{A} \in \mathcal{Q}^*(A)$. We first discuss computing the inertia of a sign-nonsingular symmetric matrix from its sign pattern. Hall, Li and Wang [3] gave several characterizations of the symmetric sign patterns that require unique inertia in 2001. Their result implies that the inertia of a sign-nonsingular symmetric matrix is determined

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 236–248, 2005.

uniquely by the sign pattern. We prove the uniqueness of the inertia of a sign-nonsingular symmetric matrix in a different way from Hall *et al.*, that is, we find a nested sequence $\{A_k\}_{k=1}^{n}$ of the principal submatrices of order k such that at least one of A_{k-1} and A_k is sign-nonsingular for any $k = 2, \ldots, n$. Our proof is based on a characterization of the structure of a symmetric bipartite graph. Furthermore, our proof naturally provides an $O(nm)$ time algorithm for computing the inertia of a sign-nonsingular symmetric matrix of order n with m nonzero entries.

We also discuss the problem of deciding whether a symmetric matrix is sign-nonsingular or not. We prove that the decision problem whether a given symmetric matrix is not sign-nonsingular is NP-complete by the result of Klee *et al.* Hence, for a symmetric matrix A in general, it is NP-complete to discern whether there exists $\tilde{A}$ in $\mathcal{Q}^*(A)$ such that the inertia of $\tilde{A}$ is different from that of A.

The present paper is organized as follows. Section 2 contains some notations and preliminaries about matrices and bipartite graphs. In Section 3, we recapitulate the inertia of a symmetric matrix in terms of linear algebra. From Sections 4 to 6, we discuss the inertia of a sign-nonsingular symmetric matrix. Section 4 gives the structure of a symmetric bipartite graph with perfect matchings. In Section 5, we show that the inertia of a sign-nonsingular symmetric matrix A is determined uniquely by the sign pattern. In Section 6, we design an efficient algorithm for computing the inertia of a sign-nonsingular symmetric matrix. In Section 7, we describe NP-completeness of the decision problem whether there exists $\tilde{A}$ in $\mathcal{Q}^*(A)$ such that the inertia of $\tilde{A}$ is different from that of A for a symmetric matrix A in general.

2 Matrices and Bipartite Graphs

For a matrix A, the *row index set* and the *column index set* are denoted by $\mathrm{Row}(A)$ and $\mathrm{Col}(A)$, i.e., $A = (a_{ij} \mid i \in \mathrm{Row}(A), j \in \mathrm{Col}(A))$, where a_{ij} is the (i, j)-entry of A. For $I \subseteq \mathrm{Row}(A)$ and $J \subseteq \mathrm{Col}(A)$, $A[I, J] = (a_{ij} \mid i \in I, j \in J)$ means the submatrix of A with row set I and column set J.

For a square matrix A of order n, the *determinant* of A is defined by

$$\det A := \sum_{\pi \in \mathcal{S}_n} \mathrm{sgn}\pi \prod_{i=1}^{n} a_{i\pi(i)}, \tag{1}$$

where $\mathcal{S}_n$ denotes the set of all the permutations of order n, and $\mathrm{sgn}\pi \in \{1, -1\}$ is the signature of the permutation $\pi \in \mathcal{S}_n$. A square matrix is said to be *nonsingular* if its determinant is distinct from zero.

As a combinatorial counterpart of nonsingularity, we say that a matrix A is *term-nonsingular* if the expansion in (1) contains at least one nonvanishing term, that is, if $a_{i\pi(i)} \neq 0$ $(\forall i \in \mathrm{Row}(A))$ for some permutation $\pi \in \mathcal{S}_n$. Since the right-hand side of (1) is distinct from zero only if the summation contains a nonzero term, nonsingularity implies term-nonsingularity. The *term-rank* of A is

the maximum size of a term-nonsingular submatrix of A. We denote term-rank of A by t-rank$A$. Hence we have t-rank$A \geq$ rank$A$.

Let $G = (U, V; E)$ be a bipartite graph with vertex sets U, V and an edge set $E \subseteq U \times V$. A *path* $P \subseteq E$ is a sequence of consecutive edges in a graph. A *circuit* $C \subseteq E$ is a path which ends at the vertex it begins. For an edge subset $F \subseteq E$, we denote by ∂F the set of all the end-vertices of edges in F, i.e., $\partial F := \{u, v \mid (u, v) \in F\}$. An edge subset M in G is called a *matching* if $2|M| = |\partial M|$, and a matching M is said to be a *perfect matching* if $\partial M = U \cup V$.

With a matrix A, we associate a bipartite graph $G(A) = (U, V; E)$ with vertex sets $U := \{u_i \mid i \in \mathrm{Row}(A)\}$ and $V := \{v_j \mid j \in \mathrm{Col}(A)\}$. The edge set E is given by $E := \{(u_i, v_j) \mid a_{ij} \neq 0, u_i \in U, v_j \in V\}$, that is, an edge of $G(A)$ represents a nonvanishing entry of A. A perfect matching in $G(A)$ corresponds to one nonvanishing term of $\det A$. Therefore, A is term-nonsingular if and only if $G(A)$ has a perfect matching. Furthermore, the term-rank of A is equal to the maximum size of a matching in $G(A)$.

For a square matrix A, we say that A has the *equisignum* determinant if every nonvanishing term of the determinant of A has the same sign. If a square matrix has the equisignum determinant, the matrix is obviously nonsingular independently of the magnitude of its entries. It is known that $\tilde{A}$ is nonsingular for any $\tilde{A} \in \mathcal{Q}(A)$ if and only if A has the equisignum determinant [5]. Because of $\mathcal{Q}^*(A) \subseteq \mathcal{Q}(A)$, this is a sufficient condition for a symmetric matrix to be sign-nonsingular.

Consider a directed bipartite graph $D = (U, V; E)$. An edge subset F is said to be *central* if the subgraph obtained from D by deleting the vertices ∂F has a perfect matching. A circuit is said to be *oddly oriented* in D if the circuit contains an odd number of edges that are directed in the direction of each orientation of the circuit. We say that a directed bipartite graph D is *Pfaffian* if every central circuit in D is oddly oriented.

Let $D(A)$ be a directed bipartite graph associated with a matrix A defined as follows. A vertex set of $D(A)$ is the same as that of $G(A)$, and an edge $e = (u_i, v_j)$ is oriented from u_i to v_j for $a_{ij} > 0$, and from v_j to u_i for $a_{ij} < 0$. It is known that A has the equisignum determinant if and only if $D(A)$ is Pfaffian [6]. The complexity status of the decision problem whether a given matrix has the equisignum determinant or not had been an open problem for a long time. This problem is polynomial time equivalent to the decision problem whether a given undirected bipartite graph has a Pfaffian orientation [10]. In 1999, Robertson, Seymour and Thomas [9] gave a polynomial time algorithm to discern if a given bipartite graph has a Pfaffian orientation. Moreover, for a bipartite graph which has a Pfaffian orientation, there exists a polynomial time algorithm to discern if a given orientation is Pfaffian or not. Thus it can be tested in polynomial time whether $\tilde{A}$ is nonsingular for any $\tilde{A} \in \mathcal{Q}(A)$ for a given matrix A. See also McCuaig [7] for a survey on these decision problems.

3 The Inertia of Symmetric Matrices

In this section, we recapitulate the inertia of a symmetric matrix in terms of linear algebra. We consider a symmetric matrix A of order n, where $\mathrm{Row}(A)$ and $\mathrm{Col}(A)$ are both identical with a finite set N of cardinality n.

Let $p(A)$ be the number of positive eigenvalues of A, $q(A)$ the number of negative eigenvalues, and $z(A)$ the number of zero eigenvalues, all counting multiplicity. The ordered triple $\mathrm{In}(A) := (p(A), q(A), z(A))$ is called the *inertia* of A. By the definition of the inertia, we have

$$\mathrm{rank}A = p(A) + q(A), \tag{2}$$

$$z(A) = n - \mathrm{rank}A. \tag{3}$$

Consider transforming A to $S^\top AS$ by a nonsingular matrix S. This transformation is called a *congruence transformation*. It is known that the inertia of a symmetric matrix is invariant under congruence transformations (Sylvester's law of inertia [4]). Hence, a symmetric matrix A is congruent to a diagonal matrix as follows:

$$S^\top AS = \mathrm{diag}(1, \ldots, 1, -1, \ldots, -1, 0, \ldots, 0), \tag{4}$$

where the indicated matrix has exactly $p(A)$ "1", $q(A)$ "-1", and $z(A)$ "0" in its diagonal entries. If A is nonsingular, it follows from (4) that

$$\det(S^\top AS) = (\det S)^2 \det A = (-1)^{q(A)}. \tag{5}$$

The sign of $\det A$ is thus determined by the parity of $q(A)$.

Let A_k be a principal submatrix of order k for $1 \leq k \leq n$. Then there exists a nonsingular matrix S such that

$$S^\top AS = \begin{pmatrix} I_{p(A_k)} & O & O \\ O & -I_{q(A_k)} & O \\ O & O & H \end{pmatrix}, \tag{6}$$

where H is a symmetric matrix, and I_p is the identity matrix of order p. This implies

$$p(A_k) \leq p(A), \quad q(A_k) \leq q(A).$$

Let $\{A_k\}_{k=1}^n$ be a nested sequence of principal submatrices of order k ($A_n := A$). In a similar way, we have

$$p(A_k) \leq p(A_l), \quad q(A_k) \leq q(A_l), \tag{7}$$

for each $1 \leq k < l \leq n$.

A nonsingular symmetric matrix A has a nested sequence $\{A_k\}_{k=1}^n$ of principal submatrices such that no two consecutive submatrices are singular. We call such a sequence *admissible*. If we find an admissible nested sequence $\{A_k\}_{k=1}^n$ of principal submatrices, then the inertia of A can be obtained by comparing the signs of $\det A_k$ as follows.

If both A_k and A_{k-1} are nonsingular ($k \geq 2$), then (5) implies that the signs of the principal minors $\det A_k$ and $\det A_{k-1}$ are equal to $(-1)^{q(A_k)}$ and $(-1)^{q(A_{k-1})}$, respectively. Moreover, the difference between $q(A_k)$ and $q(A_{k-1})$ is at most one by (2) and (7). Therefore, we have

$$\mathrm{In}(A_k) = \begin{cases} \mathrm{In}(A_{k-1}) + (1,0,0) & (\text{if } \operatorname{sgn}\det A_k = \operatorname{sgn}\det A_{k-1}) \\ \mathrm{In}(A_{k-1}) + (0,1,0) & (\text{if } \operatorname{sgn}\det A_k \neq \operatorname{sgn}\det A_{k-1}). \end{cases} \tag{8}$$

In an admissible sequence $\{A_k\}_{k=1}^n$, if A_k is nonsingular and A_{k-1} is singular ($k \geq 3$), then A_{k-2} is nonsingular. Consider transforming A_k by a nonsingular matrix S to diagonalize A_{k-2} as (6). Since A_{k-1} is singular and A_k is nonsingular, the determinant of the submatrix H in the right-hand side of (6) is negative. Hence $\operatorname{sgn}\det A_k \neq \operatorname{sgn}\det A_{k-2}$. Then, by (5), the parities of $q(A_k)$ and $q(A_{k-2})$ are different. Moreover, the difference between $q(A_k)$ and $q(A_{k-2})$ is at most two by (2) and (7). Therefore, we have

$$\mathrm{In}(A_k) = \mathrm{In}(A_{k-2}) + (1,1,0). \tag{9}$$

4 Symmetric Bipartite Graphs

Let $G = (U, V; E)$ be a bipartite graph with vertex sets $U := \{u_1, \ldots, u_n\}$ and $V := \{v_1, \ldots, v_n\}$. Both of the vertex sets are identified with $N := \{1, \ldots, n\}$. A bipartite graph $G = (U, V; E)$ is said to be *symmetric* if $e = (u_i, v_j) \in E$ implies $(u_j, v_i) \in E$ for all $e \in E$. The bipartite graph $G(A)$ associated with a symmetric matrix A is symmetric. For edge subsets F_1 and F_2 of G, we denote by $F_1 \triangle F_2$ the symmetric difference between F_1 and F_2.

Let $G = (U, V; E)$ be a symmetric bipartite graph with perfect matchings. An edge $(u_i, v_i) \in E$ is called a *diagonal* edge for $i \in N$. For an edge subset F of G, we call $F^\top := \{(u_j, v_i) \mid (u_i, v_j) \in F\}$ the *transpose* of F. If $F^\top$ coincides with F, F is called *symmetric*, otherwise *asymmetric*.

Let M be a perfect matching in G. A perfect matching M corresponds to a permutation $\pi \in \mathcal{S}_n$. Since G is symmetric, $M^\top$ is also a perfect matching in G, and $M^\top$ corresponds to the inverse permutation π^{-1}. A permutation π is expressed uniquely as a product of some cyclic permutations, that is, $\pi = \prod_{k=1}^h \sigma_k$, where h is a positive integer and σ_k is a cyclic permutation. Let $M_k \subseteq M$ be a matching in G corresponding to σ_k. Then M is the disjoint union of M_k, that is, $M = \bigcup_{k=1}^h M_k$.

Let C be a circuit in G. Since G is symmetric, the transpose $C^\top$ is also a circuit in G. A *chord* of a circuit is an edge in $E \setminus C$ having both ends on the circuit. We say that a circuit C is *chordless* if C has no chord.

Theorem 1. *Let G be a symmetric bipartite graph with perfect matchings. Then G satisfies the following* (a), (b) or (c).

(a) There exists a perfect matching M in G such that $(u_i, v_i) \in M$ for some $i \in N$.
(b) There exists a perfect matching M in G such that $(u_i, v_j) \in M$ and $(u_j, v_i) \in M$ for some distinct $i, j \in N$.
(c) The symmetric bipartite graph G is a disjoint union of some chordless symmetric circuits.

Proof. By the assumption, the symmetric bipartite graph G has a perfect matching M and its transposed perfect matching $M^\top$. Furthermore, M is the disjoint union of matchings M_k for $k = 1, \ldots, h$ corresponding to cyclic permutations.

We will show that if G does not satisfy the condition (a) or (c), then G satisfies the condition (b).

First, we claim that $M \cup M^\top$ consists of $C_k \cup C_k^\top$ for $k = 1, \ldots, h$, where C_k is a circuit (C_k may coincide with $C_k^\top$). Indeed, $M \cup M^\top$ is the disjoint union of $M_k \cup M_k^\top$ for $k = 1, \ldots, h$. Since M does not have either (u_i, v_j) or (u_j, v_i) for some $i, j \in N$ (otherwise, G satisfies (a) or (b)), $M_k \cup M_k^\top$ consists of circuits. If $|M_k| (\geq 3)$ is odd, then $M_k \cup M_k^\top$ is a symmetric circuit of length $2|M_k|$ (see Fig.1 for the case of $|M_k| = 5$). If $|M_k| (\geq 4)$ is even, then $M_k \cup M_k^\top$ consists of an asymmetric circuit of length $|M_k|$ and its transpose (see Fig.2 for the case of $|M_k| = 4$).

Suppose that there exists an asymmetric circuit C_k and its transposed circuit $C_k^\top (\neq C_k)$ in $M \cup M^\top$. Then $M' := M \triangle C_k^\top$ is also a perfect matching in G, and M' has both (u_j, v_i) and (u_i, v_j) for each $(u_i, v_j) \in C_k \cap M$. This implies G satisfies the condition (b).

Hence we may suppose there exist only symmetric circuits in $M \cup M^\top$. Since G is not a disjoint union of some chordless circuits, (i) there exists $(u_i, v_j) \in E \setminus (M \cup M^\top), u_i \in \partial C_k, v_j \in \partial C_l$ for some distinct $1 \leq k, l \leq h$, or (ii) there exists $(u_i, v_j) \in E \setminus (M \cup M^\top), u_i \in \partial C_k, v_j \in \partial C_k$ for some $1 \leq k \leq h$.

Suppose G satisfies (i). Then there also exists $(u_j, v_i) \in E \setminus (M \cup M^\top), u_j \in \partial C_l, v_i \in \partial C_k$, because of symmetry of $G(A)$. Let $P_k \subseteq C_k$ be a path from $u_i \in U \cap \partial C_k$ to $v_i \in V \cap \partial C_k$ such that both end-vertices of P_k are incident with an edge of $M \cap P_k$. Let $P_l \subseteq C_l$ be a path in a similar way. Then $C' := (P_k \cup P_l) \cup \{(u_i, v_j), (u_j, v_i)\}$ is a circuit in G. Furthermore, $M' := M \triangle C'$ is also a perfect matching in G, and M' has both (u_i, v_j) and (u_j, v_i). Moreover, it shows

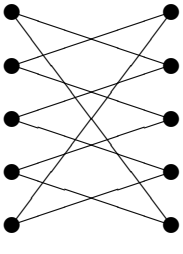


Fig. 1. The union $M_k \cup M_k^\top$ ($|M_k| = 5$)

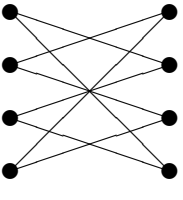


Fig. 2. The union $M_k \cup M_k^\top$ ($|M_k| = 4$)

that there exists $(u_{j'}, v_{i'}) \in (P_k^\top \cup P_l^\top) \cap M'$ for each $(u_{i'}, v_{j'}) \in (P_k \cup P_l) \cap M'$ because C_k and C_l are symmetric. Thus G satisfies the condition (b).

Suppose G satisfies (ii). Then let $P_k \subseteq C_k$ be a path from $u_i \in U \cap \partial C_k$ to $v_j \in V \cap \partial C_k$ such that both end-vertices of P_k are incident with an edge of $M \cap P_k$. Then $C' := P_k \cup \{(u_i, v_j)\}$ is a circuit in G. Furthermore, $M' := M \triangle C'$ is also a perfect matching in G, and there exists $(u_{j'}, v_{i'}) \in C_k \cap M'$ for each $(u_{i'}, v_{j'}) \in (P_k \triangle P_k^\top) \cap M'$. Thus G satisfies the condition (b).

Consequently, if G does not satisfy the condition (a) or (c), G satisfies the condition (b). □

5 Uniqueness of the Inertia

In this section, we discuss the uniqueness of the inertia of a sign-nonsingular symmetric matrix A. For convenience, we denote by $A[J]$ the principal submatrix $A[J, J]$ for $J \subseteq N$. Since sign-nonsingularity implies nonsingularity, it is clear that $z(A) = 0$.

A nested sequence $\{A_k\}_{k=1}^n$ of principal submatrices is called *sign-admissible* if at least one of A_{k-1} and A_k is sign-nonsingular for any $k = 2, \ldots, n$. It follows from (8) and (9) that if A has a sign-admissible sequence, the inertia of A is determined uniquely by the sign pattern.

However, it is not easy to find such a sequence. Even if a symmetric matrix A has the equisignum determinant, a principal submatrix of A may not be sign-nonsingular. For example,

$$A = \begin{pmatrix} +1 & +1 & +1 & -1 \\ +1 & +1 & +1 & +1 \\ +1 & +1 & -1 & 0 \\ -1 & +1 & 0 & 0 \end{pmatrix}$$

has the equisignum determinant, and hence A is sign-nonsingular, while a nested sequence of the leading principal submatrices is not sign-admissible ($A[\{1, 2\}]$ and $A[\{1, 2, 3\}]$ are not sign-nonsingular). However, the inertia is determined independently of the magnitude of its entries, $\mathrm{In}(A) = (2, 1, 0)$. This is because each of $A[\{2\}]$, $A[\{2, 3\}]$ and $A[\{1, 2, 3, 4\}]$ has the equisignum determinant.

Thus, it is sufficient to find one sign-admissible nested sequence of principal submatrices for determining the inertia uniquely by the sign pattern. To find a sign-admissible nested sequence, we will use the characterization of a symmetric bipartite graph described in Section 4.

The main result in this section is the following theorem.

Theorem 2. *Let A be a sign-nonsingular symmetric matrix. Then there exists a sign-admissible nested sequence of principal submatrices. Hence $\tilde{A}$ has the same inertia for any $\tilde{A} \in \mathcal{Q}^*(A)$.*

We first provide the following three lemmas to prove Theorem 2.

Lemma 1. *Let A be a sign-nonsingular symmetric matrix such that $G(A)$ satisfies the condition* (a) of Theorem 1, that is, there exists a perfect matching M in $G(A)$ containing (u_i, v_i) for some $i \in N$. Then the principal submatrix $A' := A[N \setminus \{i\}]$ is sign-nonsingular, and the inertia of A is obtained by

$$\mathrm{In}(A) = \begin{cases} \mathrm{In}(A') + (1,0,0) & (\text{if } a_{ii} > 0) \\ \mathrm{In}(A') + (0,1,0) & (\text{if } a_{ii} < 0). \end{cases} \tag{10}$$

Proof. The determinant of A is expanded as

$$\det A = a_{ii} \det A' + \sum_{j \neq i} (-1)^{i+j} a_{ij} \det A[N \setminus \{i\}, N \setminus \{j\}].$$

Since $G(A')$ has a perfect matching $M \setminus \{(u_i, v_i)\}$, A' is term-nonsingular. It then follows from the sign-nonsingularity of A that A' is a sign-nonsingular symmetric matrix (otherwise, the sign of the determinant of A can change by the magnitude of a_{ii}). Therefore, by (5), we have

$$\mathrm{sgn} \det A = (-1)^{q(A)} = \mathrm{sgn}(a_{ii})(-1)^{q(A')},$$

which implies that the parity of $q(A)$ depends on that of $q(A')$ and the sign of a_{ii}. Therefore, by (8), we have

$$(p(A), q(A)) = \begin{cases} (p(A') + 1, q(A')) & (\text{if } a_{ii} > 0) \\ (p(A'), q(A') + 1) & (\text{if } a_{ii} < 0). \end{cases}$$

□

Lemma 2. *Let A be a sign-nonsingular symmetric matrix such that $G(A)$ does not satisfy the condition* (a) of Theorem 1. Suppose $G(A)$ satisfies the condition (b), that is, there exists a perfect matching M in $G(A)$ such that $(u_i, v_j) \in M$ and $(u_j, v_i) \in M$ for some distinct $i, j \in N$. Then the principal submatrix $A' = A[N \setminus \{i, j\}]$ is sign-nonsingular, and the inertia of A is obtained by

$$\mathrm{In}(A) = \mathrm{In}(A') + (1,1,0). \tag{11}$$

Proof. In the same way as Lemma 1, the determinant of A is expanded as

$$\det A = -(a_{ij})^2 \det A' + \sum_{\substack{(k,l) \neq (i,j), \\ k \neq l}} \pm \det A[\{k,l\}, \{i,j\}] \det A[N \setminus \{k,l\}, N \setminus \{i,j\}].$$

Notice that $\det A[\{i,j\}] = -(a_{ij})^2$, because $G(A)$ has no perfect matching containing any diagonal edge by the assumption. Since A is sign-nonsingular and $G(A')$ has a perfect matching $M \setminus \{(u_i, v_j), (u_j, v_i)\}$, the principal submatrix A' is sign-nonsingular. Then, by (5), we have

$$\mathrm{sgn} \det A = (-1)^{q(A)} = (-1)^{q(A')+1},$$

which implies that the parity of $q(A)$ is different from that of $q(A')$. Therefore, by (9), we have

$$(p(A), q(A)) = (p(A') + 1, q(A') + 1).$$

□

Notice that it is not necessary that the principal submatrix $A[N \setminus \{i\}]$ or $A[N \setminus \{j\}]$ is sign-nonsingular.

Lemma 3. *Let A be a sign-nonsingular symmetric matrix of order $n := 2s + 1$ such that $G(A)$ is a chordless symmetric circuit. Let $2t$ be the number of the negative entries of A. Then the inertia of A is obtained by*

$$\mathrm{In}(A) = \begin{cases} (s+1, s, 0) \ (\text{if } t \equiv s \bmod 2) \\ (s, s+1, 0) \ (\text{if } t \not\equiv s \bmod 2). \end{cases} \tag{12}$$

Proof. Since $G(A)$ is a chordless symmetric circuit, the matrix size n is odd. Since $\det A$ has only two nonvanishing terms, we have

$$\operatorname{sgn} \det A = (-1)^t. \tag{13}$$

Let A' be the principal submatrix obtained by deleting the ith row and the ith column from A for any $i \in N$. Then $G(A')$ consists of two paths. Since $G(A')$ has only one perfect matching, A' has the equisignum determinant. Hence A' is sign-nonsingular. Then, by (5), we have

$$\operatorname{sgn} \det A' = (-1)^{q(A')}. \tag{14}$$

Since $G(A')$ consists of two paths and $G(A')$ has no diagonal edge, $G(A')$ satisfies the condition (b) of Theorem 1. By applying (11) repeatedly, we have $p(A') = q(A') = s$. Hence, by (8), we have

$$(p(A), q(A)) = \begin{cases} (s+1, s) & (\text{if } \operatorname{sgn} \det A = \operatorname{sgn} \det A') \\ (s, s+1) & (\text{if } \operatorname{sgn} \det A \neq \operatorname{sgn} \det A'), \end{cases}$$

which together with (13) and (14) implies (12). □

A sign-nonsingular symmetric matrix A such that $G(A)$ is a chordless circuit has a sign-admissible sequence. Indeed, by row and column permutations, A is represented as

$$A = \begin{pmatrix} 0 & a_1 & 0 & \cdots & 0 & a_n \\ a_1 & 0 & a_2 & \ddots & & 0 \\ 0 & a_2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{n-2} & 0 \\ 0 & & \ddots & a_{n-2} & 0 & a_{n-1} \\ a_n & 0 & \cdots & 0 & a_{n-1} & 0 \end{pmatrix},$$

where $a_1, \ldots, a_n$ designate nonzero entries of A. Hence any nested sequence of principal submatrices in A is sign-admissible.

We are now ready to prove Theorem 2.

Proof of Theorem 2. By Lemmas 1–3 and Theorem 1, if a symmetric matrix A of order n is sign-nonsingular, then we can find a sign-nonsingular principal submatrix of order $n-1$ or $n-2$, or $G(A)$ is the disjoint union of chordless symmetric circuits. Hence there exists a sign-admissible nested sequence of principal submatrices in A. Therefore, the inertia of A is uniquely determined by the sign pattern. □

6 Algorithm for Sign-Nonsingular Symmetric Matrices

In this section, we present a polynomial time algorithm for computing the inertia of a given sign-nonsingular symmetric matrix.

Let A be a sign-nonsingular symmetric matrix of order n. Then $z(A) = 0$. It is easy to see from Lemmas 1–3 that the inertia is obtained from the inertia of the sign-nonsingular symmetric matrix smaller than A by finding a perfect matching M in $G(A)$ which satisfies the condition (a) or (b) of Theorem 1. Thus we can compute the inertia of A recursively. However, we present a practically more efficient algorithm that computes the inertia by finding circuits in $M \cup M^{\top}$ in the same way as the proof of Theorem 1.

The algorithm starts with finding a perfect matching M_d in $G(A)$ with maximum number of diagonal edges. The index set N of the row and column set is partitioned into $J_+ := \{j \mid (u_j, v_j) \in M_d, a_{jj} > 0\}$, $J_- := \{j \mid (u_j, v_j) \in M_d, a_{jj} < 0\}$ and $J_* := \{j \mid (u_j, v_j) \notin M_d\}$. We can find M_d efficiently by the maximum weight perfect matching algorithm. Indeed, define the weight function $w : E \to \{0, 1\}$ on the bipartite graph $G(A)$ by $w_e = 1$ for each diagonal edge e and $w_e = 0$ for the other edges. Then a maximum weight perfect matching with respect to w corresponds to M_d. The maximum weight perfect matching algorithm runs in $\mathrm{O}(n(m + nW))$ time with Dial's implementation of Dijkstra's shortest path algorithm [1], where W is the largest weight in the graph and m is the number of edges. In this case, W is equal to 1. Hence it requires $\mathrm{O}(nm)$ time.

Let $\mathcal{C}$ be the set of all symmetric circuits in $M_d \cup M_d^{\top}$. If there exist a pair of symmetric circuits $C, C' \in \mathcal{C}$ connected by an edge in $E \setminus (M_d \cup M_d^{\top})$, then delete C and C' from $\mathcal{C}$. We repeat this until there are no such pair of symmetric circuits.

We denote by $C_1, \ldots, C_h$ the remaining symmetric circuits in $\mathcal{C}$. Let J_k be the index set of ∂C_k for each $k = 1, \ldots, h$, that is, $J_k := \{i \mid (u_i, v_j) \in C_k\}$. Let J_0 be the remaining indices, that is, $J_0 := J_* \setminus \bigcup_{k=1}^{h} J_k$. Then it follows from Lemmas 1 and 2 that the inertia of A is obtained by

$$\begin{aligned} p(A) &= |J_+| + \frac{1}{2}|J_0| + \sum_{k=1}^{h} p(A[J_k]), \\ q(A) &= |J_-| + \frac{1}{2}|J_0| + \sum_{k=1}^{h} q(A[J_k]). \end{aligned} \tag{15}$$

The first term in the right-hand side of (15) is obtained by applying (10) repeatedly to each $i \in J_+ \cup J_-$, and the second term by applying (11) repeatedly to each $i, j \in J_0$.

The inertia of $A[J_k]$ for $k = 1, \ldots, h$ is obtained by Lemmas 2 and 3. Indeed, if C_k has a chord, then find a chordless symmetric circuit C'_k in $G(A[J_k])$ (otherwise $C'_k := C_k$). Let J'_k be the index set of $\partial C'_k$.

Let $2t'_k$ be the number of the negative entries of $A[J'_k]$. Comparing $s'_k := (|J'_k| - 1)/2$ with t'_k, we have, by (11) and (12), that

$$(p(A[J_k]), q(A[J_k])) = \begin{cases} (s_k + 1, s_k) & (\text{if } t'_k \equiv s'_k \bmod 2) \\ (s_k, s_k + 1) & (\text{if } t'_k \not\equiv s'_k \bmod 2), \end{cases} \tag{16}$$

where $s_k := (|J_k| - 1)/2$. Thus the inertia of A can be computed from (15) and (16).

The algorithm is now summarized in Fig. 3.

1. Find a perfect matching M_d in $G(A)$ with maximum number of diagonal edges.
2. Find the set of all symmetric circuits $\mathcal{C}$ in $M_d \cup M_d^\top$.
3. Repeat the following.
   (1) If there exist a pair of symmetric circuits C and C' in $\mathcal{C}$ connected by an edge in $E \setminus (M_d \cup M_d^\top)$, then delete C and C' from $\mathcal{C}$.
4. Denote the remaining symmetric circuits in $\mathcal{C}$ by $C_1, \ldots, C_h$. Let J_k be the index set of C_k. For each connected component $G(A[J_k])$ for $k = 1, \ldots, h$, do the following.
   (1) If C_k has a chord, then find a chordless symmetric circuit C'_k in $G(A[J_k])$.
   (2) Compute the inertia of $A[J_k]$ by (16).
5. Return the inertia of A obtained by (15).

Fig. 3. Algorithm for computing the inertia of a sign-nonsingular symmetric matrix

The algorithm requires $O(mn)$ time in total. Indeed, it requires $O(nm)$ time to find a perfect matching M_d. In addition, it requires $O(m)$ time to find symmetric circuits $C_1, \ldots, C_h$ in $M_d \cup M_d^\top$, and $O(m)$ time to find a chordless symmetric circuit in $G(A[J_k])$ for $k = 1, \ldots, h$ by breadth first search.

Theorem 3. *For a sign-nonsingular symmetric matrix A of order n, the inertia of A can be computed in $O(nm)$ time, where m is the number of nonzero entries of A.*

7 Complexity of Testing Sign-Nonsingularity

In this section, we discuss the complexity status of the decision problem whether the inertia of a given symmetric matrix is uniquely determined by the sign pattern of the matrix entries. Hall, Li and Wang [3] proved the following theorem.

Theorem 4 (Hall, Li and Wang [3]). *For a given symmetric matrix A, $\tilde{A}$ has the same inertia for any $\tilde{A} \in \mathcal{Q}^*(A)$ if and only if*

$$\max\{\mathrm{rank}\tilde{A} \mid \tilde{A} \in \mathcal{Q}^*(A)\} = \min\{\mathrm{rank}\tilde{A} \mid \tilde{A} \in \mathcal{Q}^*(A)\}.$$

Furthermore, it is known that t-rank$A = \max\{\mathrm{rank}\tilde{A} \mid \tilde{A} \in \mathcal{Q}^*(A)\}$ [8]. Hence the inertia of A is uniquely determined by the sign pattern if and only if A satisfies

$$\text{t-rank}A = \min\{\mathrm{rank}\tilde{A} \mid \tilde{A} \in \mathcal{Q}^*(A)\}. \tag{17}$$

However, it is not clear how to discern efficiently if a given symmetric matrix satisfies (17) or not.

First, we suppose A is a term-nonsingular matrix of order n. Then A satisfies (17) if and only if A is sign-nonsingular. As already mentioned in Section 2, it can be tested in polynomial time whether a given symmetric matrix has the equisignum determinant. However, having the equisignum determinant is not a necessary condition for a symmetric matrix to be sign-nonsingular. For example,

$$A = \begin{pmatrix} +1 & 0 & +1 & +1 \\ 0 & +1 & +1 & +1 \\ +1 & +1 & -1 & 0 \\ +1 & +1 & 0 & -1 \end{pmatrix}$$

is sign-nonsingular, while it does not have the equisignum determinant [3].

We prove the following theorem.

Theorem 5. *For a symmetric matrix A, the problem of deciding whether A is not sign-nonsingular is NP-complete.*

To prove Theorem 5, we use the following result by Klee, Ladner and Manber [5].

Theorem 6 (Klee, Ladner and Manber [5]). *For an $n \times (n + \lfloor n^{1/k} \rfloor)$ matrix A with a positive integer k, the problem of deciding whether there exists $\tilde{A} \in \mathcal{Q}(A)$ such that it is not row full-rank is NP-complete.*

Proof of Theorem 5. It is clear that the decision problem whether a given symmetric matrix A is not sign-nonsingular is in NP, as it suffices to exhibit a singular symmetric matrix with the same sign pattern as A.

Suppose we can discern in polynomial time whether A is not sign-nonsingular. Consider

$$A = \begin{pmatrix} O & B \\ B^\top & I \end{pmatrix},$$

where B is a rectangular matrix such that $|\mathrm{Col}(B)| = 2|\mathrm{Row}(B)|$ and t-rank$B = |\mathrm{Row}(B)|$, and I is an identity matrix. Then $\det A = -\det B^\top B$ holds. Since $B^\top B$ is a positive semidefinite matrix, this implies $\det A \leq 0$. By the assumption, we can discern in polynomial time whether there exists $\tilde{A} = \begin{pmatrix} O & \tilde{B} \\ \tilde{B}^\top & \tilde{I} \end{pmatrix} \in \mathcal{Q}^*(A)$

such that $\det \tilde{A} = -\det \tilde{B}^\top \tilde{B} = 0$. This implies that we can test in polynomial time whether there exists $\tilde{B} \in \mathcal{Q}(B)$ which is not row full-rank. Hence it is NP-complete to discern whether A is not sign-nonsingular by Theorem 6. □

It follows from Theorem 5 that the decision problem whether the inertia of A is not determined by the sign pattern is also NP-complete.

Corollary 1. *For a symmetric matrix A, it is NP-complete to discern whether there exists $\tilde{A}$ in $\mathcal{Q}^*(A)$ such that the inertia of $\tilde{A}$ is different from that of A.*

Acknowledgements

The authors are grateful to Kazuo Murota for his useful comments and valuable discussions.

References

1. R. K. Ahuja, T. L. Magnanti and J. B. Orlin: *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, 1993.
2. R. A. Brualdi and B. L. Shader: *Matrices of Sign-solvable Linear Systems*, Cambridge University Press, Cambridge 1995.
3. F. J. Hall, Z. Li, and D. Wang: Symmetric sign pattern matrices that require unique inertia, *Linear Algebra and Its Applications*, 338, pp.153–169, 2001.
4. R. A. Horn and C. R. Johnson: *Matrix Analysis*, Cambridge University Press, 1985.
5. V. Klee, R. Ladner and R. Manber: Sign-solvability revisited, *Linear Algebra and Its Applications*, 59, pp.131–158, 1984.
6. L. Lovász and M. D. Plummer: *Matching Theory*, *Annals of Discrete Math*, 29, North-Holland, Amsterdam, 1986.
7. W. McCuaig: Pólya's permanent problem, *The Electronic Journal of Combinatorics*, 11, # R79, 2004.
8. K. Murota: An identity for bipartite matching and symmetric determinant, *Linear Algebra and Its Applications*, 222, pp.261–274, 1995.
9. N. Robertson, P. D. Seymour, and R. Thomas: Permanents, Pfaffian orientations, and even directed circuits, *Annals of Mathematics*, 150, no. 3, pp.929–975, 1999.
10. V. V. Vazirani and M. Yannakakis: Pfaffian orientations, 0-1 permanents, and even cycles in directed graphs, *Discrete Applied Mathematics*, 25, pp.179–190, 1989.

Randomized Relaxation Methods for the Maximum Feasible Subsystem Problem

Edoardo Amaldi[1,⋆], Pietro Belotti[1], and Raphael Hauser[2,⋆⋆]

[1] DEI, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{amaldi, belotti}@elet.polimi.it
[2] Oxford University Computing Laboratory, Parks Road, Oxford OX1 3QD, UK
hauser@comlab.ox.ac.uk

Abstract. In the Max FS problem, given an infeasible linear system $A\mathbf{x} \geq \mathbf{b}$, one wishes to find a feasible subsystem containing a maximum number of inequalities. This NP-hard problem has interesting applications in a variety of fields. In some challenging applications in telecommunications and computational biology one faces very large Max FS instances with up to millions of inequalities in thousands of variables. We propose to tackle large-scale instances of Max FS using randomized and thermal variants of the classical relaxation method for solving systems of linear inequalities. We present a theoretical analysis of one particular version of such a method in which we derive a lower bound on the probability that it identifies an optimal solution within a given number of iterations. This bound, which is expressed as a function of a condition number of the input data, implies that with probability 1 the randomized method identifies an optimal solution after finitely many iterations. We also present computational results obtained for medium- to large-scale instances arising in the planning of digital video broadcasts and in the modelling of the energy functions driving protein folding. Our experiments indicate that these methods perform very well in practice.

1 Introduction

We consider the following problem, referred to as Max FS:

Given an infeasible linear system $A\mathbf{x} \geq \mathbf{b}$, where $A \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$, find a feasible subsystem containing as many inequalities as possible.

⋆ E. A. and P. B. were partially supported through grant 2003014039 from Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR), Italy.

⋆⋆ R. H. was supported through grant NAL/00720/G from the Nuffield Foundation and through grant GR/M30975 from the Engineering and Physical Sciences Research Council of the UK.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 249–264, 2005.

This problem has a number of interesting applications in various fields such as operations research [8, 13], radiation therapy [14], image and signal processing [16], statistical discriminant analysis and machine learning [5, 15]. There is growing interest in MAX FS due to the fact that many complex phenomena that can be well-approximated with linear models yield formulations involving large and generally infeasible linear systems for which approximate solutions in terms of l_1- or l_2-norms are not meaningful. Note that since linear system feasibility can be checked in polynomial time, the MAX FS structure differs substantially from that of MAX SAT, the well-known problem of satisfying a maximum number of Boolean clauses. Like MAX SAT though, MAX FS can be approximated within a factor of 2 but it does not admit a polynomial-time approximation scheme, unless P = NP [3]. The reader is referred to [1] for a survey of work on MAX FS up to 2003.

Apart from large-scale instances arising in classification problems [5, 15] and in the context of infeasible linear programs [8, 13], very large MAX FS instances with up to millions of inequalities and thousands of variables occur in several challenging applications in telecommunications (digital video broadcasting [22]) and computational biology (modelling the energy function that drives the folding of proteins [17]) that are of current interest. Such instances are beyond the reach of available computational approaches, which include state-of-the-art MIP solvers applied to big-M formulations, the best available heuristics [8], as well as the latest polyhedral techniques based on partial set covering formulations [4, 20] and combinatorial Benders' cuts [9].

To tackle large-scale instances of MAX FS, we propose to use randomized and thermal variants of the classical Agmon-Motzkin-Schoenberg (AMS) relaxation method for solving systems of linear inequalities. Deterministic AMS relaxation versions have been extensively studied in the mathematical programming literature, often as special cases of subgradient methods (see e.g. [12, 23]), and in the machine learning literature under the name of perceptron procedures (see e.g. [6, 18]). The algorithms we propose and investigate here extend the thermal perceptron heuristic [11]. It is worth pointing out that the power of randomization in relaxation/subgradient type methods, which was first exploited by the machine learning community, has also been recently studied in the context of *feasible* systems of convex inequalities, see [21], and in the context of the minimization of sums of *convex* component functions with a large number of summands, see [19]. In this paper the focus is on a difficult combinatorial optimization problem.

In Section 2 we give a detailed description of the methods under consideration. Section 3 is then devoted to a theoretical analysis of one particular variant. In Section 4 we report some computational results that show that our methods are very competitive for solving medium- to large-scale MAX FS instances.

2 Randomized and Thermal Relaxation Methods

The *relaxation method* is a simple algorithmic framework for solving systems of linear inequalities, see e.g. [12, 23]. At every iteration the attention is focused

on reducing the violation of a single inequality. Given a feasible system $A\mathbf{x} \geq \mathbf{b}$ with $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$, as well as a starting point $\mathbf{x}_0 \in \mathbb{R}^n$, such a method generates a sequence $\left(\mathbf{x}_i\right)_{\mathbb{N}} \subset \mathbb{R}^n$ of approximate solutions as follows. In iteration i an index $k_i \in \{1,\ldots,m\}$ is selected according to some specified rule, the corresponding inequality $\mathbf{a}_{k_i}\mathbf{x} \geq b_{k_i}$ is considered, and the next iterate is determined via

$$\mathbf{x}_{i+1} = \begin{cases} \mathbf{x}_i + \eta_i\ \mathbf{a}_{k_i} & \text{with probability } p_i \text{ if } \mathbf{a}_{k_i}\mathbf{x}_i < b_{k_i} \\ \mathbf{x}_i & \text{otherwise,} \end{cases} \tag{1}$$

where $\eta_i > 0$ and $p_i \in [0,1]$ may be chosen differently in every iteration. When $\mathbf{x}_{i+1} \neq \mathbf{x}_i$ we say that an *update* has occurred. Any specific variant of this approach can be characterized by

i) a *selection rule* which specifies the way a single inequality is selected at each iteration,

ii) a *step-length multiplier rule* which specifies how the sequence of η_i is determined,

iii) and a *decision rule* characterized by the updating probability p_i.

Let us now briefly comment on some specific choices of these rules.

i) In most classical versions of the relaxation method the inequalities are considered in cyclic order or according to the largest violation. In the *randomized* relaxation method, we assume that k_i is chosen uniformly at random from the set $\{1,\ldots,m\}$. This introduces a first level of randomization.
ii) Classical choices for the *step length multiplier* η_i include $\eta_i \equiv 1$ in the *perceptron method* and $\eta_i = (b_{k_i} - \mathbf{a}_{k_i}\mathbf{x}_i)/\|\mathbf{a}_{k_i}\|$ in the *cyclic projection method*. In the algorithms we consider in this paper, we use $\eta_i > 0$ as a tool for prioritizing the level of attention paid to attempting to satisfy the different inequalities depending on their varying degrees of violation.
iii) The *updating probability* p_i with which an update is carried out introduces a second layer of randomization. When $p_i < 1$, we therefore speak of a *doubly randomized* method, while $p_i \equiv 1$ corresponds to the standard (singly) randomized relaxation method. The use of double randomization is interesting because it allows replacing a large number of small updates by updates of moderate size that occur with small probability.

To adapt the relaxation method to the Max FS framework and infeasible systems $A\mathbf{x} \geq \mathbf{b}$, we make η_i and p_i dependent on a parameter t_i that keeps changing over time. This leads to the so-called *thermal* variants of the randomized relaxation method, henceforth referred to as RTR. The motivation is that the choice of η_i should initially allow large corrections to the iterates $\mathbf{x}_i$, in order for the algorithm to make good progress. However, once the iterates have moved to a region of the solution space containing good approximate solutions of Max FS, $\mathbf{x}_i$ should be further improved only by small local updates, as large updates risk destroying much of the progress previously achieved. Therefore, we want to force the method to favour updates generated by inequalities with progressively smaller violation $v_i = \max\{0, b_{k_i} - \mathbf{a}_{k_i}\mathbf{x}_i\}$. A natural way to achieve this is by

introducing a decreasing *temperature schedule* $(t_i)_{\mathbb{N}} \subset \mathbb{R}_+$ and to choose $(\eta_i)_{\mathbb{N}}$ as a function of t_i and v_i, for example, by setting

$$\eta_i = \frac{t_i}{t_0} \exp(-v_i/t_i). \tag{2}$$

For large values of t_i, any violated inequality yields a significant update, while for small t_i only inequalities with small violations v_i yield significant updates. In the doubly randomized framework one trades small step-length multipliers η_i for small updating probabilities p_i. For example, in the RTR variant analyzed in Section 3, we choose $\eta_i = 1$ for all i and

$$p_i = \frac{t_i}{t_0} \exp(-v_i/t_i). \tag{3}$$

Like in simulated annealing (SA), t_i can be interpreted as a temperature that gradually "cools down" the activity level of the algorithm and forces it to pursue more localized searches. It is however important to point out that the similarity does not extend beyond this intuitive level. For example, in doubly randomized RTR methods the updating probability depends only on the violation v_i of a single inequality, while in SA it would depend on the objective function value (i.e., the number of satisfied inequalities) of the new iterate $\mathbf{x}_{i+1}$. The two approaches are thus significantly different. Our analysis of Section 3 is based on a technique that does not occur in the SA literature and our probabilistic finite termination results are stronger than SA asymptotic convergence guarantees.

We conclude with two technical remarks:

- To complete the description of our RTR methods, we need to specify the termination criterion. Since these methods do not ever properly terminate for infeasible systems, the best iterate found up to iteration i needs to be stored in a so-called *ratchet vector* $\mathbf{z}_i$. In practical implementations, a maximum number of iterations, *maxit*, is selected at the outset and the ratchet vector at iteration *maxit* is returned as an approximate solution of Max FS.
- The inherent parallelism of the RTR methods can be exploited to tackle very large Max FS instances such as those considered in Section 4. See also [7] for parallel versions of relaxation methods for linear (convex) feasible systems.

3 Probabilistic Termination Guarantees

In this section we establish, as an illustrative example, the probabilistic finite termination guarantee to an optimal solution of Max FS for a particular RTR variant. The method we consider is the doubly randomized RTR variant characterized by the choice $\eta_i \equiv 1$ and p_i as in (3). We investigate the situation where this method is applied to a homogeneous system $A\mathbf{x} > \mathbf{0}$ of strict inequalities in which all row vectors $\mathbf{a}$ of A have unit length.

We write $\mathcal{A}$ for the set of row vectors of the input matrix A. For any $\mathbf{x} \in \mathbb{R}^n$ we write $\mathcal{A}_\mathbf{x} := \{\mathbf{a} \in \mathcal{A} : \mathbf{a} \cdot \mathbf{x} > 0\}$ for the subset of inequalities satisfied by $\mathbf{x}$ and $\#\mathcal{A}_\mathbf{x}$ for its cardinality.

Definition 1. *Let $(t_i)_{\mathbb{N}_0}$ be a decreasing temperature schedule, $\mathbf{x}_0 \in \mathbb{R}^n$ a starting point with unit norm, and $\mathbf{z}_0 = \mathbf{x}_0$. Let $(\mathbf{A}_i)_{\mathbb{N}_0}$ be a sequence of i.i.d. random vectors which are uniformly distributed over $\mathcal{A}$. Then a sequence $(U_i)_{\mathbb{N}_0}$ of binary random variables and sequences $(\mathbf{X}_i)_{\mathbb{N}_0}$ and $(\mathbf{Z}_i)_{\mathbb{N}_0}$ of random vectors in $\mathbb{R}^n$ are defined via the following recursive relations:*

i) $\mathbf{X}_0 \equiv x_0$, $\mathbf{Z}_0 \equiv \mathbf{z}_0$,

ii) U_i *is* $\sigma(\mathbf{A}_i, \mathbf{X}_i)$*-measurable binary random variable for which*

$$\mathrm{P}\big[U_i = 1 \,\big\|\, \mathbf{A}_i \cdot \mathbf{X}_i \le 0\big] = p_i := \frac{t_i}{t_0}\,\exp\Big(-\frac{|\mathbf{A}_i \cdot \mathbf{X}_i|}{t_i}\Big)$$
$$\mathrm{P}\big[U_i = 1 \,\big\|\, \mathbf{A}_i \cdot \mathbf{X}_i > 0\big] = 0.$$

iii) $\mathbf{X}_{i+1} = \mathbf{X}_i + U_i\,\mathbf{A}_i$,

iv) $\mathbf{Z}_{i+1} = \mathbf{X}_{i+1}$ *if* $\#\mathcal{A}_{\mathbf{X}_{i+1}} > \#\mathcal{A}_{\mathbf{Z}_i}$, *and* $\mathbf{Z}_{i+1} = \mathbf{Z}_i$ *otherwise.*

One run of our algorithm consists in sampling the random sequences $(U_i)_{\mathbb{N}_0}$ and $(\mathbf{a}_i)_{\mathbb{N}_0}$, thus defining sample sequences $(\mathbf{x}_i)_{\mathbb{N}_0}$ and $(\mathbf{z}_i)_{\mathbb{N}_0}$ of $(\mathbf{X}_i)_{\mathbb{N}_0}$ and $(\mathbf{Z}_i)_{\mathbb{N}_0}$ respectively. In order to analyze the probabilistic behaviour of our algorithm, it therefore suffices to analyze the random sequences $(\mathbf{X}_i)_{\mathbb{N}_0}$ and $(\mathbf{Z}_i)_{\mathbb{N}_0}$. In Theorem 1 we derive a lower bound on the probability that $\mathbf{Z}_i$ becomes Max FS-optimal in at most a certain number of iterations. Furthermore, we will show that almost surely all but finitely many terms of the ratchet sequence $(\mathbf{Z}_i)_{\mathbb{N}_0}$ are Max FS-optimal solutions.

The general line of attack can be outlined as follows. Let $\tilde{\mathbf{x}}$ be an optimal solution of the Max FS problem, that is, $\#\mathcal{A}_{\tilde{\mathbf{x}}} = m^* := \max\{\#\mathcal{A}_{\mathbf{x}} : \mathbf{x} \in \mathbb{R}^n\}$, and let $\mathbf{x}^* := \tilde{\mathbf{x}}/\alpha$, where $\alpha := \min\{\mathbf{a} \cdot \tilde{\mathbf{x}} : a \in \mathcal{A}_{\tilde{\mathbf{x}}}\}$. We focus our attention on this fixed optimal solution $\mathbf{x}^*$ and the associated feasible subsystem $A_{\mathbf{x}^*}\mathbf{x} > \mathbf{0}$. As long as the iterate $\mathbf{x}_i$ satisfies $\#\mathcal{A}_{\mathbf{x}_i} < m^*$, there exists at least one $\mathbf{a} \in \mathcal{A}_{\mathbf{x}^*}$ for which $\mathbf{a} \cdot \mathbf{x}_i \le 0$. Under uniform sampling this particular inequality is drawn with probability $1/m$, and then $\mathbf{x}_{i+1} \neq \mathbf{x}_i$ with probability p_i. According to Lemma 1 below, no more than a certain number $\tilde{\gamma}$ of consecutive such updates can occur before $\mathbf{x}_i$ satisfies $\mathbf{a} \cdot \mathbf{x}_i > 0$ for all $\mathbf{a} \in \mathcal{A}_{\mathbf{x}^*}$, and then $\mathbf{x}_i$ is optimal for Max FS. Subdividing the sequence of iterates into epochs of length $\tilde{\gamma}$, optimality is achieved with positive probability in each epoch. Although the law of large numbers can be seen as the motivating idea of our approach, the analysis is more complicated because the events that optimality is achieved in iterations i and $i+1$ respectively are not independent of one another.

The convergence speed claimed in Theorem 1 is a function of a condition number $\kappa(A)$, defined in [2] as the inverse of the smallest possible strictly positive k-dimensional volume of a parallelepiped generated by a subset of k row vectors of A for k ranging from 1 to m. In the case where A consists of rational data of total bit length $\mathscr{L}$, the inequality $\kappa(A) \le 2^{\mathscr{L}}$ holds, see [2] for details. Here we also write $\kappa(\mathcal{A})$, since the condition number does not depend on the ordering of the input vectors.

Theorem 1. *Let* $M := 2\kappa(\mathcal{A})2^{3n/2}+1$ *and* $\gamma := (M+\|\mathbf{x}^*\|)^2$. *If the temperature schedule in Definition 1 satisfies* $t_i \geq \frac{c}{\log i}$ *for* $i \geq 2$ *and* $t_i \geq \frac{c}{\log 2}$ *for* $i = 0, 1$ *for some constant* $c \geq 2M\gamma$, *then the following statements hold true.*

i) For all $k \in \mathbb{N}$, *we have*

$$\mathrm{P}\big[\#\mathcal{A}_{\mathbf{Z}_{k\gamma}} = m^*\big] \geq 1 - \prod_{j=2+\lfloor \frac{r}{\gamma} \rfloor}^{k} \Big(1 - \frac{1}{j\gamma}\Big),$$

where $r > 0$ *is large enough so that* $\frac{m \log x}{\log 2} < x^{\frac{1}{2\gamma}}$ *holds for all* $x > r$.

ii) Almost surely the ratchet vector becomes Max FS*-optimal after a finite number of iterations.*

We need two lemmas in the proof of Theorem 1. The first one deals with the finite termination of the relaxation method for feasible inequality systems.

Lemma 1. *Let any variant of the relaxation method be applied to a feasible linear system* $A\mathbf{x} > 0$ *and let* $(i(l))_{\mathbb{N}} \subset \mathbb{N}$ *be the sequence of iterations in which updates occur. If* $\lim_{u\to\infty} \sum_{l=1}^{u} \eta_{i(l)}(2-\eta_{i(l)}) = +\infty$ *then* $\mathbf{x}_{i(l)}$ *becomes a feasible solution for some* $l \leq \tilde{\gamma} := \max\big\{u \in \mathbb{N} : \sum_{l=1}^{u} \eta_{i(l)}(2-\eta_{i(l)}) \leq (\|\mathbf{x}_0\| + \|\mathbf{x}^*\|)^2\big\}$.

Note that $\tilde{\gamma} = \tilde{\gamma}(\|\mathbf{x}_0\|, \|\mathbf{x}^*\|, (\eta_{i(l)})_{\mathbb{N}})$ is a monotone increasing function of $\|\mathbf{x}_0\|$ if all other entries are fixed. Lemma 1 can easily be shown by adapting the proof of a similar result from [6, 18] that was given for constant $\eta_i \equiv 1$. It has long been known that the number of updates required can grow exponentially in the input size [23], but we note that a variant of the relaxation method in which the underlying space is periodically rescaled was recently shown to be polynomial in a probabilistic framework [10].

In the case of infeasible systems, infinitely many updates occur due to violated inequalities. However, our second lemma shows the interesting fact that under bounded step-length multipliers the sequence of iterates $(\mathbf{x}_i)_{\mathbb{N}}$ remains bounded.

Lemma 2. *Let* $A\mathbf{x} > 0$ *be an arbitrary homogeneous system,* $\mathbf{x}_0 \in \mathbb{R}^n$ *and* $(\mathbf{x}_i)_{i\in\mathbb{N}}$ *the sequence of iterates generated by any version of the relaxation method that uses a bounded sequence of step-length multipliers* $0 < \eta_i \leq \overline{\eta}$ *for some* $\overline{\eta} > 0$. *Then for all* i, *we have*

$$\|\mathbf{x}_i\| \leq M(\mathcal{A}, \mathbf{x}_0) := 2\max\Big\{\|\mathbf{x}_0\|\ ,\ \overline{\eta}\big(\kappa(\mathcal{A})\, 2^{3n/2} + 1\big)\Big\}.$$

For a proof see Theorem 6.1 in [2]. Lemma 2 is a generalization of the classical *perceptron boundedness theorem* [6, 18] to non-constant step-length multipliers, but its proof is based on a different approach in order to settle the open question of expressing M as an explicit function of the input data $(\mathcal{A}, \mathbf{x}_0)$, and in order to overcome a limitation of the classical approach by which a lower bound on the η_i had to be assumed. The explicit expression for M is a crucial ingredient in the proof of Theorem 1 which we are now ready to present.

Proof. (Theorem 1) Under the assumptions $\|\mathbf{a}\| = 1 \ \forall\, \mathbf{a} \in \mathcal{A}$ and $\|\mathbf{x}_0\| = 1$, Lemma 2 implies that $\|\mathbf{X}_i\| \leq M$ for all $i \in \mathbb{N}$. The monotonicity of $\tilde{\gamma}$ implies that

$$\tilde{\gamma}\big(\|\mathbf{X}_{k\gamma}\|, \|\mathbf{x}^*\|, (1)_{\mathbb{N}}\big) \leq \gamma, \quad \forall k \in \mathbb{N}. \tag{4}$$

Since $(1)_{\mathbb{N}}$ is shift-invariant, we can concentrate on epochs of constant length γ under the general framework outlined above.

i) Denoting the complement of a set $\mathcal{B} \subseteq \mathcal{A}$ by $\mathcal{B}^c = \mathcal{A} \setminus \mathcal{B}$, we define the events $E_i := \{\#\mathcal{A}_{\mathbf{X}_i} = m^*\} \cup \big\{\mathbf{A}_i \in \mathcal{A}_{\mathbf{x}^*} \cap \mathcal{A}^c_{\mathbf{X}_i}, U_i = 1\big\}$. We claim that for all $k \in \mathbb{N}$,

$$E^{[k]} := E_{k\gamma-1} \cap \cdots \cap E_{(k-1)\gamma} \subseteq \big\{\#\mathcal{A}_{\mathbf{Z}_{k\gamma}} = m^*\big\}. \tag{5}$$

Indeed, suppose that $E^{[k]}$ occurs. If $\#\mathcal{A}_{\mathbf{X}_i} = m^*$ for some index $i \in [(k-1)\gamma, k\gamma - 1]$ then the ratchet will assure that $\#\mathcal{A}_{\mathbf{Z}_i} = m^*$ and a fortiori $\#\mathcal{A}_{\mathbf{Z}_{k\gamma}} = m^*$. Otherwise for γ consecutive iterations a violated inequality $\mathbf{A}_i \cdot \mathbf{X}_i \leq 0$ with $\mathbf{A}_i \in \mathcal{A}_{\mathbf{x}^*}$ is drawn and $\mathbf{X}_i$ is updated to $\mathbf{X}_{i+1} = \mathbf{X}_i + \mathbf{A}_i$. But then (4) and Lemma 1 imply that $\mathcal{A}_{\mathbf{X}_i} = \mathcal{A}_{\mathbf{x}^*}$ occurs for some $i \in [(k-1)\gamma + 1, k\gamma]$ and it is again the case that $\#\mathcal{A}_{\mathbf{Z}_{k\gamma}} = m^*$, establishing the truth of our claim.

Equation (5) implies

$$\mathrm{P}\left[\#\mathcal{A}_{\mathbf{Z}_{k\gamma}} < m^* \,\middle\|\, \#\mathcal{A}_{\mathbf{Z}_{(k-1)\gamma}} < m^*\right] \leq 1 - \mathrm{P}\left[E^{[k]} \,\middle\|\, \#\mathcal{A}_{\mathbf{Z}_{(k-1)\gamma}} < m^*\right]. \tag{6}$$

Our next goal is to bound the right-hand side of (6). As long as $\#\mathcal{A}_{\mathbf{X}_i} < m$, we have $\mathcal{A}_{\mathbf{x}^*} \cap \mathcal{A}^c_{\mathbf{X}_i} \neq \emptyset$, so that for any $\sigma(\mathbf{X}_0; \mathbf{A}_0, \ldots, \mathbf{A}_{i-1}; U_0, \ldots, U_{i-1})$-measurable event $F \subseteq \{\#\mathcal{A}_{\mathbf{X}_i} < m^*\}$ it is true that

$$\mathrm{P}\big[E_i \,\big\|\, F\big] = \mathrm{P}\big[\mathbf{A}_i \in \mathcal{A}_{\mathbf{x}^*} \cap \mathcal{A}^c_{\mathbf{X}_i}, U_i = 1 \,\big\|\, F\big] \geq \frac{1}{m} \times \frac{t_i}{t_0} \exp\left(-\frac{|\mathbf{A}_i \cdot \mathbf{X}_i|}{t_i}\right). \tag{7}$$

In particular, $B_i \cap \{\#\mathcal{A}_{\mathbf{X}_i} < m^*\}$ is $\sigma(\mathbf{X}_0; \mathbf{A}_0, \ldots, \mathbf{A}_{i-1}; U_0, \ldots, U_{i-1})$-measurable for $i > (k-1)\gamma$, where $B_i := \big(\bigcap_{j=(k-1)\gamma}^{i-1} E_j\big) \cap \big\{\#\mathcal{A}_{\mathbf{Z}_{(k-1)\gamma}} < m^*\big\}$. Therefore,

$$\begin{aligned}
\mathrm{P}\big[E_i \,\big\|\, B_i\big] &= \mathrm{P}\big[E_i \,\big\|\, \#\mathcal{A}_{\mathbf{X}_i} < m^*, B_i\big] \times \mathrm{P}\big[\#\mathcal{A}_{\mathbf{X}_i} < m^* \,\big\|\, B_i\big] \\
&\qquad + \mathrm{P}\big[E_i \,\big\|\, \#\mathcal{A}_{\mathbf{X}_i} = m^*, B_i\big] \times \mathrm{P}\big[\#\mathcal{A}_{\mathbf{X}_i} = m^* \,\big\|\, B_i\big] \\
&\overset{(7)}{\geq} \frac{t_i}{mt_0} \exp\left(-\frac{|\mathbf{A}_i \cdot \mathbf{X}_i|}{t_i}\right) \times \mathrm{P}\big[\#\mathcal{A}_{\mathbf{X}_i} < m^* \,\big\|\, B_i\big] \\
&\qquad + 1 \times \mathrm{P}\big[\#\mathcal{A}_{\mathbf{X}_i} = m^* \,\big\|\, B_i\big] \\
&\geq \frac{t_i}{mt_0} \exp\left(-\frac{|\mathbf{A}_i \cdot \mathbf{X}_i|}{t_i}\right),
\end{aligned}$$

so that

$$\mathrm{P}\big[E^{[k]} \,\big\|\, \#\mathcal{A}_{\mathbf{Z}_{(k-1)\gamma}} < m^*\big] \;= \tag{8}$$

$$\mathrm{P}\big[E_{(k-1)\gamma} \,\big\|\, \#\mathcal{A}_{\mathbf{Z}_{(k-1)\gamma}} < m^*\big] \;\times \prod_{i=(k-1)\gamma+1}^{k\gamma-1} \mathrm{P}\big[E_i \,\big\|\, B_i\big]$$

$$\geq \prod_{i=(k-1)\gamma}^{k\gamma-1} \frac{t_i}{mt_0}\, \exp\left(-\frac{|\mathbf{A}_i \cdot \mathbf{X}_i|}{t_i}\right). \tag{9}$$

We wish to bound this quantity solely in terms of the epoch k. The assumption on the temperature schedule implies $\frac{|\mathbf{A}_i \cdot \mathbf{X}_i|}{t_i} < \frac{M \log i}{2M\gamma} = \frac{\log i}{2\gamma}$, and hence

$$\frac{t_i}{mt_0}\, \exp\left(-\frac{|\mathbf{A}_i \cdot \mathbf{X}_i|}{t_i}\right) > \frac{\log 2}{m \log i} \times i^{-\frac{1}{2\gamma}}.$$

Let $r > 0$ be such that $\frac{m \log x}{\log 2} < x^{\frac{1}{2\gamma}}$ for all $x > r$. Then for $i > r$ we have $\frac{t_i}{mt_0} \exp\big(-\frac{|\mathbf{A}_i \cdot \mathbf{X}_i|}{t_i}\big) > i^{-\frac{1}{\gamma}}$. Using this inequality in (9), it follows that for $k > 1 + \frac{r}{\gamma}$

$$\mathrm{P}\big[E^{[k]} \,\big\|\, \#\mathcal{A}_{\mathbf{Z}_{(k-1)\gamma}} < m^*\big] > \prod_{i=(k-1)\gamma}^{k\gamma-1} i^{-\frac{1}{\gamma}} > \big((k\gamma)^{-\frac{1}{\gamma}}\big)^{\gamma} = \frac{1}{k\gamma}.$$

This finally provides the desired bound on the right-hand side of (6), so that

$$\mathrm{P}\big[\#\mathcal{A}_{\mathbf{Z}_{k\gamma}} < m^* \,\big\|\, \#\mathcal{A}_{\mathbf{Z}_{(k-1)\gamma}} < m^*\big] \leq 1 - \frac{1}{k\gamma}. \tag{10}$$

Using (10), we now find

$$\mathrm{P}\big[\#\mathcal{A}_{\mathbf{Z}_{k\gamma}} < m^*\big] \;=\; \mathrm{P}\big[\#\mathcal{A}_{\mathbf{Z}_0} < m^*\big] \times \prod_{j=1}^{k} \mathrm{P}\big[\#\mathcal{A}_{\mathbf{Z}_{j\gamma}} < m^* \,\big\|\, \#\mathcal{A}_{\mathbf{Z}_{(j-1)\gamma}} < m^*\big]$$

$$\leq \prod_{j=2+\lfloor \frac{r}{\gamma} \rfloor}^{k} \left(1 - \frac{1}{j\gamma}\right), \tag{11}$$

which settles the claim of part i).

ii) Since

$$\lim_{k\to\infty} \log \prod_{j=2+\lfloor \frac{r}{\gamma} \rfloor}^{k} \left(1 - \frac{1}{j\gamma}\right) \leq - \lim_{k\to\infty} \sum_{j=2+\lfloor \frac{r}{\gamma} \rfloor}^{k} \frac{1}{j\gamma} = -\infty,$$

the right-hand side of (11) converges to zero when $k \to \infty$, and since the events $\{\#\mathcal{A}_{\mathbf{Z}_i} < m^*\}$ are nested, we have

$$\mathrm{P}\left[\bigcap_{i\in\mathbb{N}} \{\#\mathcal{A}_{\mathbf{Z}_i} < m\}\right] = \lim_{k\to\infty} \mathrm{P}\left[\#\mathcal{A}_{\mathbf{Z}_{k\gamma}} < m\right] = 0. \qquad \square$$

We note that the proof of Theorem 1 can be modified to obtain similar probabilistic finite termination results for other variants of the RTR method applied to the Max FS problem. In particular, for the RTR variant with $p_i = \exp(-v_i/t_i)$ and $\eta_i = t_i/t_0$ additional difficulties arise since p_i and η_i jointly decrease.

4 Computational Results

In our theoretical analysis, we have considered long non-overlapping sequences of updates and very slow temperature decreasing schedules. By weakening the mutual dependence of the events of finding an optimal solution during particular epochs, we made it possible to use arguments akin to the law of large numbers. But our theoretical bounds on the temperature schedules are overly cautious and, as we shall see, our RTR methods perform very well in practice with much faster linearly decreasing temperature schedules.

To tackle large-scale instances of Max FS, we have developed an efficient RTR implementation which includes the following additional features.

Linearly Decreasing Temperature Schedule. We have devised a simple way to select the initial temperature based on the average inequality violation: t_0 is renormalized after randomly picking m inequalities as $t_0 := \frac{1}{3}t_0 + \frac{2}{3}\sum_{k=1}^{m} v_k$. We also use an adaptive way of decreasing t_i over the maximum number of iterations *maxit*, which is effective on infeasible systems coming from a variety of applications: $t_i := \left(1 - \frac{i}{maxit}\right) t_0$, where i is the iteration index.

Projection on Bounding Box. When facing Max FS instances whose variables are subject to mandatory lower and upper bounds, the next iterate obtained after each update is projected back into the bounding box whenever it falls out of it.

Update Direction Based on a Block of Inequalities. Good sub-optimal solutions are obtained more rapidly by considering a block-iterative variant of the RTR method in which the update direction is given by a (convex) combination of the $\mathbf{a}_k$'s of the violated inequalities in a block and the size of the block is decreased over the iterations. For the block idea applied to feasible linear systems see for example [7]. Suppose that $I \subseteq \{1, \ldots, m\}$ is the index set of the inequalities in the block that are violated by the current iterate $\mathbf{x}_i$. Since in our experiments all inequalities are considered to be equally important, v_i can be replaced by $\frac{\sum_{k\in I}(b_k - \mathbf{a}_k\mathbf{x}_i)}{|I|}$ and the updating step is described by

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \eta_i \frac{\sum_{k\in I} \mathbf{a}_k}{|I|}, \tag{12}$$

where $|I|$ denotes the number of violated inequalities in the block. Decreasing the size of the block to 1 during the iterations (proportionally to t_i and to the number of inequalities violated by the current iterate) leads to substantial improvements.

Variable-Based Local Search. After a certain number of non-improving iterations of plain RTR, we apply a local search step with respect to single variables. Given a current solution $\mathbf{x} = (x_1, x_2 \ldots, x_n)$, we look for a variable index j, $1 \leq j \leq n$, and a scalar δ such that the number of inequalities satisfied by $(x_1, x_2 \ldots, x_j + \delta \ldots, x_n)$ is maximum. This 1-D search, which has a complexity of nm, yields improved solutions. Encouraged by the performance of the RTR block variant, we have adopted a grouping strategy also for this variable-based local search step, which provides better quality solutions.

One-Dimensional Optimization Along the Update Direction. At each iteration, after $\mathbf{x}_{i+1}$ is found, we look for a point of the segment connecting $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, i.e., of the set $\{\mathbf{x}_i + \alpha(\mathbf{x}_{i+1} - \mathbf{x}_i),\ 0 \leq \alpha \leq 1\}$, that satisfies the largest number of inequalities. Despite the higher complexity (nm if we are using the group subgradient or the variable-based local search), this procedure yields improved solutions for a number of instances.

Preprocessing Step. If the variables are subject to mandatory upper and lower bounds, some inequalities can be satisfied by either all or none of the points lying in the bounding box, and hence may be deleted. By taking into account the sign of the coefficients a_{ij} as well as the bounds on the variables, we can easily determine whether any given inequality can be satisfied by all $\mathbf{x}$ in the bounding box or by none of them. Similarly, by checking the coefficients of a given variable x_j it is easy to establish whether its value can be fixed to the upper or lower bound, provided the vector $\mathbf{b}$ is suitably modified. The two above preprocessing steps are repeated alternatively until no more inequalities or variables can be deleted from the system.

If the variables are bounded, the initial iterate $\mathbf{x}_0$ is generated uniformly at random within the bounding box, otherwise it is taken as $\mathbf{0}$. The RTR method terminates either after *maxit* iterations or when no significant improvement is achieved during a predefined number of iterations.

We now present some computational results obtained for large-scale Max FS instances arising in digital video broadcasting [22] and in modelling protein folding potentials [17]. We also report some results for small-size instances occurring in two-group linear discriminant analysis. The experiments were carried out with a 2.8 GHz processor and 2 GB of RAM.

Digital Video Broadcasting (DVB). In planning a DVB network, an interesting problem is that of determining the emission power of a given set of n transmitters so as to maximize territory coverage [22]. Suppose the territory is subdivided into m sufficiently small squared areas, considered as test points (TPs). A signal emitted from a transmitter is useful or interfering at a TPdepending on the arrival delay. In the single frequency and fixed window model [22], for each TP i the signal quality constraint can be linearized as follows:

$$\sum_{j=1}^{n} a_{ij}\ x_j \geq b_i, \tag{13}$$

where the variable x_j indicates the emission power of the jth transmitter, subject to a mandatory upper bound given by the maximum power $p_{\max}$. The field strength a_{ij} of the signal arriving at TP i from transmitter j is positive (negative) for the useful (interfering) part of the signal, and b_i is the minimum field strength required to cover TP i with probability 0.95. Since total coverage is usually not achievable, one faces sparse instances of Max FS with one inequality per TP, a variable per transmitter and mandatory upper (maximum power) and lower bounds for each variable. A reasonable discretization yields instances for the whole Italian territory with of the order of 55000 inequalities in a few thousands of variables. To maximize the population covered, a weight can be assigned to each inequality (13) and one searches for a Feasible Subsystem (FS) of maximum total weight. See [22] for the modelling details.

Computational experiments were conducted on a selection of DVB instances with different sizes m and n. In the upper part of Table 1 we report the number of inequalities in the feasible subsystems obtained with our RTR method and the computing time needed to find them. The results and performance are compared with Cplex 8.1 MIP solver applied to the big-M formulations. For some instance we mention the best solutions obtained at different times. Although a reasonable value of M is easily derived from the variable bounds, it is quite large due to the problem nature and it makes the linear relaxation very poor. The time limit of two hours allowed Cplex to solve optimally only the smallest instances and one medium-size instance. According to Table 1, the approximate solutions provided by the RTR method are often of much better quality than the best solutions found by Cplex in two hours (and mentioned in the table with the time to find them). Moreover, RTR typically finds a good solution almost immediately and improves it subsequently. As an example, for instance *dvb5* RTR yields a feasible subsystem with 3346 inequalities after 0.16 sec. and improves it to size 3354 within 2.33 sec., while Cplex yields a feasible subsystem with 3314 inequalities after 41.13 sec. and improves it to size 3336 after 77089.65 sec. It is worth pointing out that, because of the wide range of values the system coefficients can take, the implementation of the best available (LP-based) heuristic [8] based on MINOS and the exact method presented in [9] failed out of numerical problems on all these instances.

Protein Folding Potentials. As described in [17], the problem of modelling the energy function (potential) underlying the folding of amino acid sequences into proteins gives rise to large and dense linear systems with millions up to tens of millions inequalities in hundreds of variables. Let $E(\mathbf{s}, \mathbf{t})$ denote the energy of sequence $\mathbf{s}$ when folded into the three-dimensional structure $\mathbf{t}$ and let $\mathbf{t}_s^*$ be the native structure of sequence $\mathbf{s}$. The premise that the energy of the native structure is lower than that of any structure chosen from a set of "decoys" amounts to

$$E(\mathbf{s}, \mathbf{t}) - E(\mathbf{s}, \mathbf{t}_s^*) > 0 \quad \forall \mathbf{s},\ \forall \mathbf{t} \neq \mathbf{t}_s^*. \tag{14}$$

Focusing on the common features, potential modelling can then be formalized as the problem of approximating the unknown energy E by a linear combination of some appropriate basis function set $\{\phi_i(\mathbf{s}, \mathbf{t})\}_{1 \leq i \leq n}$, i.e., by $\tilde{E}(\mathbf{s}, \mathbf{t}) =$

Table 1. Comparison between the RTR method and Cplex 8.1 MIP solver applied to the big-M formulations for some DVB and protein folding instances. A " " indicates a proven optimal solution, "–" that no feasible solution has been found within the time limit, and "†" that the corresponding instance is feasible

			RTR		CPLEX	
			FS size	time in sec.	FS size	time in sec.
DIGITAL VIDEO BROADCASTING						
dvb1	550	487	538*	0.01	537	0.02
					538*	0.36
dvb2	912	487	867	0.01	850	0.02
			870*	19.58	870*	14.28
dvb3	1580	487	1531	2.09	1515	2.00
			1533	13.99	1534*	24.38
dvb4	3767	487	3346	0.16	3314	41.13
			3354	2.33	3336	77089.65
dvb5	6649	487	5757	46.88	–	–
dvb6	14057	487	12819	128.38	12365	58.20
dvb7	11229	487	9475	1.16	9480*	36.97
dvb8	19916	487	17154	167.49	15754	246.75
dvb9	8701	487	7479	12.09	–	–
dvb10	48758	487	41884	110.94	40082	1562.43
dvb11	49008	1681	42740	159.10	–	–
PROTEIN FOLDING POTENTIALS						
prot1	19404	12	16772	1.22	–	–
prot2	92890	212	92383	242.80	–	–
prot3	250233	150	249865	9.11	–	–
			250225	92.85	–	–
prot4	349584	22	268396	121.53	–	–
prot5	837802	11	837689	23.88	–	–
feas1	200176	301	200173	21.21	†	
feas2	240219	301	240213	27.55	†	
feas3	280364	301	280364*	32.92	†	
feas4	320543	301	320541	30.29	†	
feas5	360630	301	360629	31.85	†	
feas6	401115	301	401112	75.50	†	

$\sum_{i=1}^{n} x_i \phi_i(\mathbf{s}, \mathbf{t})$ where $\mathbf{x}$ is the vector of parameters. As a simple example, $\phi_i(\mathbf{s}, \mathbf{t})$ may count the number of contacts between a certain pair of amino acids that appear when $\mathbf{s}$ is folded into structure $\mathbf{t}$. Thus, by requiring that the potential models $\tilde{E}$ satisfy (14), we have $\sum_{i=1}^{n} x_i(\phi_i(\mathbf{s}, \mathbf{t}) - \phi_i(\mathbf{s}, \mathbf{t}_\mathbf{s}^*)) > 0$ for all $\mathbf{s}$ and all $\mathbf{t} \neq \mathbf{t}_s^*$. Since perfect structure recognition is unrealistic, these dense homogeneous systems are generally infeasible and one is interested in finding an $\mathbf{x}$ satisfying as many inequalities (14) as possible [17].

As mentioned in the lower part of Table 1, these instances are much larger than the DVB ones, and the corresponding big-M formulations could not be solved within the time limit of two hours and did not provide any primal solution.

For three out of the five first instances, our RTR variant yields in a very short amount of computing time a feasible subsystem containing more than 99.4% of the inequalities in the original system, that is, a solution which is within 0.6% from the optimum. Note, for example, that for *prot3* the feasible subsystem with 249865 inequalities found by the RTR variant after 9 sec. is improved to size 250225, with only 8 violated inequalities, just after 92.85 sec.

To further evaluate the quality of the RTR solutions, we have also considered five feasible systems with up to 400000 inequalities in 301 variables. For such systems, the Max FS-optimal solution is obviously known and can be used for comparison purposes. It is worth emphasizing that, unlike linear programming solvers, our RTR variants deal with such instances (feasible subsystems) as if they were infeasible and strive to find an iterate satisfying as many inequalities as possible. The results are reported in the lower part of Table 1, where a "†" indicates that the MIP solver performance on the big-M formulations is of no interest for these feasible systems. Our RTR method turns out to yield extremely good solutions (within 0.001% from the optimum) in less than 80 sec., even for the largest instance with 401115 inequalities. Note that, on the same PC, Cplex 8.1 MIP solver could not even start optimizing the big-M formulation for the feasible instances with more than 300000 inequalities due to excessive memory requirements.

Discriminant Analysis. To assess the performance of the RTR method on smaller instances with known optimal solutions, we have considered the collection of linear classification instances used in [9]. The general problem is to determine the values of the parameters of a hyperplane so as to separate two groups of data points as well as possible. In Table 2 RTR results are compared with those obtained with the exact method based on Combinatorial Benders' Cuts (cbc) [9] and with Cplex 8.1 MIP solver applied to the big-M formulations. The focus here is on the instances with more than 300 inequalities. We have adopted a simple two-phase approach: after running a given number of RTR iterations (5000), we consider the inequalities in the largest feasible subsystem found, F, as mandatory and we apply a MIP solver to the reduced big-M formulation which aims at identifying among the other inequalities those that are consistent with F.

The upper part of Table 2 corresponds to the instances that are solved optimally with both the cbc method and the big-M formulations, the middle part to those solved only by cbc, and the lower part to the instances that could not be solved by either of them within the time limit of 10000 sec. For the cbc method and the big-M formulations, we report either the optimal solution and the computing time to find it (if it is reached within the time limit) or the best lower bound. For the RTR method, we report the value of the best solution and the time needed to find it. Note that, for the instances that are easily solved by cbc, the RTR variant yields near-optimal solutions within less than 7 seconds. For the instances that could not be solved to optimality within the time limit,

Table 2. Comparison between the two-phase RTR variant, the CBC method and Cplex 8.1 MIP solver applied to the big-M formulations for some small classification instances. A "–" means that the time limit has been reached, whereas " " indicates proven optimal solutions. Computing times are expressed in seconds

			RTR+		CPLEX		CBC	
			FS size	time	FS size	time	FS size	time
BCW-367	367	10	359*	1.31	359*	365	359*	1
BCW-683	683	10	669	6.13	673*	6750	673*	10
BusVan445	445	18	436	4.76	436	–	437*	102
Bv-os-376	376	18	363	5.50	367	–	368*	125
Solar-flare-323	323	12	284	1.03	282	–	285*	3
Flags-169	169	29	159	1.24	159	–	159	–
Horse-colic-253	253	26	240	7.60	240	–	240	–
Horse-colic-185	183	26	173	3.08	173	–	173	–
Solar-flare-1066	1066	12	817	6.56	796	–	782	–
			823	139.27				

RTR compares well with the two exact approaches since it provides the best solutions in extremely short computing times.

5 Concluding Remarks

We have proposed randomized and thermal variants of the classical relaxation method to tackle large-scale instances of Max FS. We have sketched a framework to analyze the probabilistic termination properties of these RTR methods in terms of the input data. In particular, we have established sufficient conditions under which one of these methods is guaranteed to find with probability 1 an optimal solution of Max FS within a finite number of iterations. Computational results obtained for instances arising in the planning of digital video broadcasts and in the modelling of protein folding potentials indicate that a simple RTR variant performs very well in practice. Interestingly it provides a remarkably good trade-off between solution quality and computing time even for small linear classification instances.

As our RTR methods do not require computationally expensive matrix operations such as matrix inversion, they are particularly suited to large-scale Max FS instances. We are currently investigating ways to exploit the inherent parallelism to tackle protein folding instances with up to forty millions of inequalities.

Acknowledgments

The authors would like to thank John Chinneck (Gianni Codato and Matteo Fischetti) for trying to tackle the DVB instances with his (their) LP-based heuristic

(exact algorithm), and Fabrizio Rossi and Stefano Smriglio (Michael Wagner and Jarek Meller) for providing the DVB (protein folding) data.

References

1. E. Amaldi. The maximum feasible subsystem problem and some applications. In A. Agnetis and G. Di Pillo, editors, *Modelli e Algoritmi per l'ottimizzazione di sistemi complessi*. Pitagora Editrice Bologna, 2003.
2. E. Amaldi and R. Hauser. Boundedness theorems for the relaxation method. Under minor revision for *Mathematics of Oper. Res.*, available from Optimization Online.
3. E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147:181–210, 1995.
4. E. Amaldi, M. E. Pfetsch, and L. E. Trotter Jr. On the maximum feasible subsystem problem, IISs and IIS-hypergraphs. *Math. Programming A*, 95:533–554, 2003.
5. K. P. Bennett and E. Bredensteiner. A parametric optimization method for machine learning. *INFORMS Journal on Computing*, 9:311–318, 1997.
6. H. D. Block and S. A. Levin. On the boundedness of an iterative procedure for solving a system of linear inequalities. In *Proceedings of AMS*, pages 229–235, 1970.
7. Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, algorithms and applications*. Oxford University Press, 1997.
8. J. Chinneck. Fast heuristics for the maximum feasible subsystem problem. *INFORMS Journal on Computing*, 13:210–213, 2001.
9. G. Codato and M. Fischetti. Combinatorial Benders' cuts. In *Proceedings of IPCO*, volume 3064 of *LNCS*, pages 178–195. Springer, 2004.
10. J. Dunagan and S. Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. In *Proceedings of STOC*, pages 315–320. ACM Press, 2004.
11. M. Frean. A "thermal" perceptron learning rule. *Neural Comp.*, 4(6):946–957, 1992.
12. J. L. Goffin. The relaxation method for solving systems of linear inequalities. *Mathematics of Oper. Res.*, 5:388–414, 1980.
13. H. J. Greenberg and F. H. Murphy. Approaches to diagnosing infeasible linear programs. *ORSA Journal on Computing*, 3:253–261, 1991.
14. E. K. Lee, R. J. Gallagher, and M. Zaider. Planning implants of radionuclides for the treatment of prostate cancer: An application of MIP. *Optima*, 61:1–7, 1999.
15. O. Mangasarian. Machine learning via polyhedral concave minimization. In H. Fischer et al., editor, *Applied Mathematics and Parallel Computing*, pages 175 – 188. Physica-Verlag, 1996.
16. M. Mattavelli, V. Noel, and E. Amaldi. Fast line detection algorithms based on combinatorial optimization. In *Proceedings of IWVF*, LCNS 2059, pages 410–419. Springer, 2001.
17. J. Meller, M. Wagner, and R. Elber. Solving huge linear programming problems for the design of protein folding potentials. *Math. Programming B*, 101:301–318, 2004.
18. M. L. Minsky and S. Papert. *Perceptrons: An introduction to computational Geometry*. MIT Press, Cambridge, MA, 1988. Expanded edition.

19. A. Nedić and D. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM J. on Optimization*, 12:109–138, 2001.
20. M. E. Pfetsch. *The maximum feasible subsystem problem and vertex-facet incidences of polyhedra*. PhD thesis, Dep. of Mathematics, Technische Universität Berlin, October 2002.
21. B. T. Polyak. Random algorithms for solving convex inequalities. In D. Butnariu et al., editor, *Inherently parallel algorithms in feasibility and other applications*. Elsevier, 2001.
22. F. Rossi, A. Sassano, and S. Smriglio. Models and algorithms for terrestrial digital broadcasting. *Ann. of Oper. Res.*, 107(3):267–283, 2001.
23. A. Schrijver. *Theory of Linear and Integer Programming*. Wiley & Sons, 1986.

On Clique Separators, Nearly Chordal Graphs, and the Maximum Weight Stable Set Problem

Andreas Brandstädt[1] and Chính T. Hoàng[2]

[1] Institut für Informatik, Universität Rostock, D-18051 Rostock, Germany
ab@informatik.uni-rostock.de
[2] Department of Physics and Computing, Wilfrid Laurier University, Waterloo, Ontario, Canada
choang@wlu.ca

Abstract. Clique separators in graphs are a helpful tool used by Tarjan as a divide-and-conquer approach for solving various graph problems such as the Maximum Weight Stable Set (MWS) Problem, Coloring and Minimum Fill-in but few examples are known where this approach was used. We combine decomposition by clique separators and by homogeneous sets and show that the resulting binary tree gives an efficient way for solving the MWS problem. Moreover, we combine this approach with the concept of nearly chordal and nearly perfect and obtain some new graph classes where MWS is solvable in polynomial time by our approach. On some of these classes, the unweighted Maximum Stable Set (MS) Problem was known to be solvable in polynomial time by the struction method or by augmenting techniques, respectively, but the complexity of the MWS problem was open. A graph is *nearly chordal* if for each of its vertices, the subgraph induced by the set of its nonneighbors is chordal, and analogously for nearly perfect graphs.

1 Introduction

For a graph $G = (V, E)$ and a vertex weight function w on V, let $\alpha_w(G)$ ($\alpha(G)$) denote the maximum weight (maximum cardinality) of a stable vertex set in G.

The *Maximum Weight Stable* (or *Independent*) *Set* (*MWS*) Problem asks for a stable set of maximum weight in the given graph G with vertex weight function w. The *MS problem* is the MWS problem if all vertices v have the same weight $w(v) = 1$.

The MWS problem is one of the fundamental algorithmic graph problems occuring in many models of optimization and computer science in its original form or in variants such as the Minimum Vertex Cover Problem or modifications.

It is known to be NP-complete in general and solvable in polynomial time on various graph classes by various techniques. Some of these techniques such as augmenting, reducing α-redundant vertices and struction lead only to efficient solutions of the unweighted Maximum Stable Set (MS) Problem.

A famous divide-and-conquer approach by using clique separators is described by Tarjan in [22]. It works for various problems on graphs such as Minimum fill-

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 265–275, 2005.

in, Coloring, Maximum Clique, and the MWS problem as shown in [22]. The subgraphs not containing clique separators are called *atoms* in [22]. Whenever MWS is efficiently solvable on the atoms of a graph G, it is efficiently solvable on G. However, few examples are known where this approach could be applied for obtaining a polynomial time MWS algorithm on a graph class. A recent example is given by Alekseev [1] showing that for a certain subclass $\mathcal{C}$ of P_5-free graphs, the atoms are $3K_2$-free from which it follows that the MWS problem is solvable in polynomial time on this graph class $\mathcal{C}$.

We combine decomposition by clique separators with decomposition by homogeneous sets in order to refine graph decomposition, and we use the following notion: Let Π denote a graph property. A graph is *nearly* Π if for each of its vertices, the subgraph induced by the set of its nonneighbors has property Π. We deal with the cases $\Pi \in \{\text{chordal, perfect}\}$.

Obviously, the MWS problem on a graph G with vertex weight function w can be reduced to the same problem on antineighborhoods of vertices in the following way:

$$\alpha_w(G) = \max\{w(v) + \alpha_w(G[\overline{N}(v)]) \mid v \in V\}$$

Thus, whenever MWS is solvable in time T on a class with property Π, it is solvable on nearly Π graphs in time nT. For example, Frank [12] gave a linear time algorithm for the MWS problem on chordal graphs. Hence, the MWS problem can be solved in time $\mathcal{O}(nm)$ for nearly chordal graphs. Grötschel, Lovász and Schrijver [14] gave a polynomial time algorithm for the MWS problem on perfect graphs. Thus, the MWS problem can be solved in polynomial time for nearly perfect graphs.

We combine these two examples with our decomposition by clique separators and homogeneous sets in order to give new classes where MWS can be solved in polynomial time. In particular, we can explain some results in a simpler way which were originally obtained by struction introduced by Ebenegger, Hammer and de Werra [11] (a complicated method to compute the unweighted stability number of a graph which came from optimizing Boolean functions), we can extend them to MWS and we use clique separators in combination with nearly chordal graphs for the MWS problem on the following two classes where the complexity of the MWS problem remained open:

(*i*) for (P_6, C_4)-free graphs which considerably extends and improves the polynomial time result for the MS problem given in [20] by using augmenting techniques;
(*ii*) for (P_5, P)-free graphs which extends and improves a result in [6, 17] where a (robust, see [6]) polynomial time solution for the MS problem on (P_5, P)-free graphs was given using α-redundant vertices in [6] and an augmenting argument in [17], respectively.

The class of (P_6,C_4)-free graphs contains split graphs and (C_4,$2K_2$)-free graphs, and (P_5,P)-free graphs also generalize various interesting graph classes such as split graphs, (C_4,$2K_2$)-free graphs, co-bipartite graphs, cographs and

P_4-sparse graphs (for the definition of all these classes see e.g. [5]). In Theorem 5, we show that for (P_6,C_4)-free graphs without clique separator, the MWS problem is efficiently solvable. This leads to a polynomial time MWS algorithm on this class by Tarjan's approach.

2 Basic Notions

Throughout this note, let $G = (V, E)$ be a finite undirected graph without self-loops and multiple edges and let $|V| = n$, $|E| = m$. Let $V(G) = V$ denote the vertex set of graph G. For a vertex $v \in V$, let $N(v) = \{u \mid uv \in E\}$ denote the (*open*) *neighborhood* of v in G, let $N[v] = \{v\} \cup \{u \mid uv \in E\}$ denote the (*closed*) *neighborhood* of v in G, and for a subset $U \subseteq V$ and a vertex $v \notin U$, let $N_U(v) = \{u \mid u \in U, uv \in E\}$ denote the *neighborhood* of v with respect to U. The *antineighborhood* $\overline{N}(v)$ is the set $V \setminus N[v]$ of vertices different from v which are nonadjacent to v.

Disjoint vertex sets X, Y *form a join*, denoted by $X ① Y$ (*co-join*, denoted by $X ⓪ Y$) if for all pairs $x \in X$, $y \in Y$, $xy \in E$ ($xy \notin E$) holds. We will also say that X has a join to Y, that there is a join between X and Y, or that X and Y are connected by join (and similarly for co-join). Subsequently, we will consider join and co-join also as operations, i.e., the co-join operation for disjoint vertex sets X and Y is the disjoint union of the subgraphs induced by X and Y (without edges between them), and the join operation for X and Y consists of the co-join operation for X and Y followed by adding all edges $xy \in E$, $x \in X$, $y \in Y$.

A vertex $z \in V$ *distinguishes* vertices $x, y \in V$ if $zx \in E$ and $zy \notin E$ or $zx \notin E$ and $zy \in E$. We also say that *a vertex z distinguishes a vertex set $U \subseteq V$*, $z \notin U$, if z has a neighbor and a non-neighbor in U. A vertex set $M \subseteq V$ is a *module* if no vertex from $V \setminus M$ distinguishes two vertices from M, i.e., every vertex $v \in V \setminus M$ has either a join or a co-join to M.

A graph G is *prime* if it contains only trivial modules, i.e., $\emptyset$, $V(G)$ and one-elementary vertex sets. A nontrivial module is called a *homogeneous set*. The notion of module plays a crucial role in the *modular* (or *substitution*) *decomposition* of graphs (and other discrete structures) which is of basic importance for the design of efficient algorithms - see e.g. [19] for modular decomposition of discrete structures and its algorithmic use and [18] for a linear-time algorithm constructing the modular decomposition tree of a given graph.

For $U \subseteq V$, let $G[U]$ denote the subgraph of G induced by U. Throughout this paper, all subgraphs are understood to be induced subgraphs. Let $\mathcal{F}$ denote a set of graphs. A graph G is *$\mathcal{F}$-free* if none of its induced subgraphs is in $\mathcal{F}$.

A vertex set $U \subseteq V$ is *stable* (or *independent*) in G if the vertices in U are pairwise nonadjacent. For a given graph with vertex weights, the Maximum Weight Stable Set (MWS) Problem asks for a stable set of maximum vertex weight.

Let co-$G = \overline{G} = (V, \overline{E})$ denote the *complement graph* of G. A vertex set $U \subseteq V$ is a *clique* in G if U is a stable set in $\overline{G}$. Let K_ℓ denote the clique

with ℓ vertices, and let ℓK_1 denote the stable set with ℓ vertices. K_3 is called *triangle*.

A *clique separator* or *clique cutset* in G is a clique Q such that $G[V \setminus Q]$ has more connected components than G.

For $k \geq 1$, let P_k denote a chordless path with k vertices and $k-1$ edges, and for $k \geq 3$, let C_k denote a chordless cycle with k vertices and k edges. A *hole* is a C_k with $k \geq 5$, and an *antihole* is $\overline{C}_k$ with $k \geq 5$. An *odd hole* (*odd antihole*, respectively) is a hole (antihole, respectively) with odd number of vertices.

For a subgraph H of G, a vertex not in H is a *k-vertex for H* if it has exactly k neighbors in H. We also say that H *has no k-vertex* if there is no k-vertex for H. For a set $S \subseteq V(H)$ with $|S| = k$ let M_S be the set of k-vertices for H adjacent to vertices in S. We also write $M_{a,b}$ respectively M_x for $S = \{a, b\}$ respectively $S = \{x\}$, etc. The subgraph H *dominates* the graph G if there is no 0-vertex for H in G. Let N denote the set of 0-vertices for H.

A graph is *chordal* if it contains no induced C_k, $k \geq 4$. A graph is *nearly chordal* if for each of its vertices, the subgraph induced by the set of its non-neighbors is a chordal graph.

More generally, if Π is a graph property then a graph is *nearly Π* if for each of its vertices, the subgraph induced by the set of its nonneighbors has the property Π. Note that this notion appears in the literature in many variants, e.g., as nearly bipartite graphs [3].

3 Connected (Claw, Antena, Net)-Free Graphs Are Nearly Chordal

A graph is (*claw,antena,net*)*-free* (*CAN-free*) if it contains none of the graphs in Figure 1 as induced subgraphs.

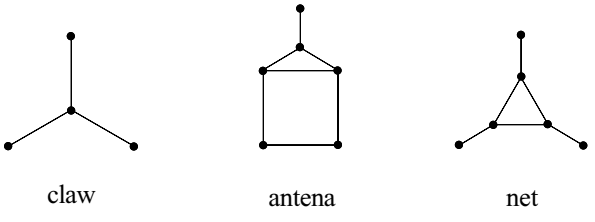


Fig. 1. The claw, antena and net

Theorem 1. *Connected* (*claw, antena, net*)*-free graphs are nearly chordal.*

Proof. Assume that there is a vertex $v \in V$ such that $G_v := G[V \setminus N[v]]$ is not chordal.

Case 1. G_v contains C_4. Let C be a C_4 in G_v with vertices v_1, v_2, v_3, v_4 and edges $v_i v_{i+1}$, $i \in \{1, 2, 3, 4\}$ (index arithmetic modulo 4). Since G is claw-free, C has no 1-vertex, and for any 2-vertex x of C, its neighbors in C are consecutive.

Since G is antena-free, no 0-vertex is adjacent to a 2-vertex of C, and since G is claw-free, no 0-vertex is adjacent to a 3- or 4-vertex of C. Thus, C has no 0-vertex since G is connected but v is a 0-vertex for C - contradiction.

Case 2. G_v is C_4-free but contains C_k for $k \geq 5$. Let C be a C_k in G_v with vertices $v_1, \ldots, v_k$ and edges $v_i v_{i+1}$, $i \in \{1, \ldots, k\}$ (index arithmetic modulo k). As before, C has no 1-vertex, and 2- and 3-vertices of C have consecutive neighbors in C. 0-vertices are not adjacent to 2-vertices since G is net-free, and 0-vertices are not adjacent to k-vertices, $k \geq 3$ since G is claw-free. Thus, C has no 0-vertex but v is a 0-vertex of C - contradiction.

This shows that G is nearly chordal. □

Corollary 1. *The MWS problem can be solved in time $\mathcal{O}(nm)$ on (claw, antena, net)-free graphs.*

This simplifies and extends a result by Hammer, Mahadev and de Werra in [15] solving the MS problem in polynomial time by so-called *struction* (a quite complicated stability reduction method) - CAN-free graphs have been one of the key examples for the use of struction. Another key example for the struction method was the larger class of (claw, net)-free graphs for which struction was shown to solve the MS problem in polynomial time in [16]; in [4], also this case was generalized in a similar way to a $\mathcal{O}(nm)$ algorithm for the MWS problem on the larger class of nearly (claw,AT)-free graphs. In the next section, we will show that (claw,net)-free graphs are nearly perfect. A third example is the class of circular-arc graphs for which struction solves the MS problem [13]; note, however, that circular-arc graphs are nearly interval graphs and thus, MWS can be solved in time $\mathcal{O}(nm)$ on circular-arc graphs (note that interval graphs are chordal.)

It seems to be a challenging task to find examples where struction works well and cannot be replaced by such a simple technique. As already mentioned, for nearly perfect graphs, MWS is solvable in polynomial time. In the next sections, we give further examples for nearly perfect graphs as well as for nearly chordal graphs.

4 Connected (Claw,Net)-Free Graphs Are Nearly Perfect

Theorem 2. *Connected (claw,net)-free graphs are nearly hole-free.*

Proof. Let G be a connected (claw,net)-free graph. Suppose there is a vertex x such that in the non-neighborhood of x there is a hole C with vertices $v_1, v_2, \ldots, v_k$ and edges $v_i v_{i+1}$ for $i = 1, \ldots, k$ and $k \geq 5$ (with subscript taken modulo k). Since G is connected, we may assume x has a neighbor y that has neighbors in C. If y has more than two neighbors in C, then y is center of a claw. If y has exactly one neighbor v_i in C, then v_i is center of a claw. So, y has exactly two consecutive neighbors in C, say v_i and v_{i+1}. But now the vertices $v_{i-1}, v_i, v_{i+1}, v_{i+2}, x, y$ form a net. □

Theorem 3. *Connected (claw,net)-free graphs are nearly odd-antihole-free.*

Proof. Let G be a connected (claw,net)-free graph. Suppose there is a vertex x such that in the non-neighborhood of x there is an odd antihole C with vertices $v_1, v_2, \ldots, v_k$ and non-edges $v_i v_{i+1}$ for $i = 1, \ldots, k$ and k being an odd integer at least seven (with subscript taken modulo k.) Since G is connected, we may assume x has a neighbor y that has neighbors in C.

First, let us remark that y cannot be adjacent to two consecutive vertices v_i, v_{i+1} of C, for otherwise y is center of a claw. Now, let v_i be a neighbor of y in C. The above remark shows that y is not adjacent to v_{i-1}, v_{i+1}. If y is not adjacent to v_{i+2}, then y must be adjacent to v_{i+3} (for otherwise, v_i, v_{i+2}, v_{i+3}, y form a claw); but now the vertices $x, y, v_i, v_{i+1}, v_{i+2}, v_{i+3}$ form a net. We have shown that if y is adjacent to v_i, then y is adjacent to v_{i+2} and nonadjacent to v_{i+1}. It follows that k is even, a contradiction. □

Corollary 2. *Connected (claw,net)-free graphs are nearly perfect.*

Proof. By Theorem 2, prime (claw,net)-free graphs are nearly hole-free, and by Theorem 3, these graphs are nearly odd-antihole-free. Thus, by the Strong Perfect Graph Theorem by Chudnovsky, Robertson, Seymour and Thomas [7], these graphs are nearly perfect. □

Corollary 3. *The Maximum Weight Stable Set Problem can be solved in polynomial time for (claw,net)-free graphs.*

5 (P_6,C_4)-Free Graphs and Clique Separators

By using augmenting techniques, Mosca [20] has shown that the MS problem can be solved in time $\mathcal{O}(n^4)$ on (P_6,C_4)-free graphs. The complexity of the weighted MWS problem remained open in [20]. Here we show that the concept of clique separators can also be applied to (P_6,C_4)-free graphs. As a first step, we show:

Theorem 4. *(P_6,C_4)-free graphs without clique cutset are nearly perfect.*

Proof. Let G be a (P_6,C_4)-free graph without clique cutset. Then G contains no C_k for $k \geq 7$ and no $\overline{C}_k$ for $k \geq 6$.

Now we use again the Strong Perfect Graph Theorem. In order to show that (P_6,C_4)-free graphs without clique cutset are nearly perfect, it is sufficient to show that these graphs are nearly C_5-free.

Assume to the contrary that there is a vertex v in G whose antineighborhood $\overline{N}(v)$ contains a C_5 C, say with vertices $v_1, \ldots, v_5$ and edges $v_i v_{i+1}$, $i \in \{1, \ldots, 5\}$ (index arithmetic modulo 5).

Since G is (P_6,C_4)-free, no vertex $x \in N(v)$ is a k-vertex, $k \in \{1, 2, 4\}$, for C, and 3-vertices have consecutive neighbors in C. Let M_k denote the set of k-vertices for C, and let $M_k(v)$ denote the set of k-vertices for C in $N(v)$, $k \in \{0, 3, 5\}$.

Since G is C_4-free, M_5 is a clique and has a join to the set M_3 of 3-vertices for C. Moreover, $M_3(v)$ is a clique since for any two vertices $x, x' \in M_3(v)$, there is a vertex $y \in C$ that is adjacent to both x and x' since x, x' have three neighbors in C. Now, $xx' \in E$ for otherwise v, x, x', y induce a C_4.

Let $Q := M_5 \cup M_3(v)$ which is a clique as shown above. Since G has no clique cutset, there is a path between v and C avoiding Q, i.e., there is a 0-vertex $x \in N(v)$ having a neighbor $y \in \overline{N}(v)$ but every such neighbor y must be a 5- or 0-vertex for C since G is P_6-free, and if y is a 5-vertex then $y \in Q$. The same argument holds for 0-vertex y and a neighbor $y' \in \overline{N}(v)$ of y which is closer to C. Thus, the vertex v together with the 0-vertex neighbors of v on one hand and C on the other hand are in different components of $G[V \setminus Q]$ - contradiction. □

Since MWS is solvable in polynomial time for perfect graphs [14], Tarjan's clique separator approach implies:

Corollary 4. *The MWS problem for (P_6, C_4)-free graphs is solvable in polynomial time.*

A more detailed structure analysis of (P_6, C_4)-free graphs is given by the following Theorem 5. To this purpose, we define specific graphs as follows. Let a *clique-C_6* denote the result of substituting cliques into the vertices of a C_6. Figure 2 shows the Petersen graph B which is the corresponding extension of the C_6 by four other vertices. Let B' denote the graph resulting from B by adding a universal vertex to B, i.e., a vertex which is adjacent to all vertices of B. A graph is *specific* if it results from substituting cliques of arbitrary size into vertices of the graph B' or is an induced subgraph of this one (obtained by substituting some cliques of size 0).

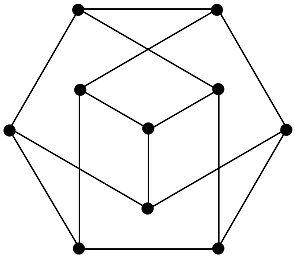


Fig. 2. The Petersen graph B

Theorem 5. *(P_6,C_4)-free graphs without clique cutset are either nearly chordal or specific.*

For space limitations, we skip the proof of Theorem 5. The time bounds given in [22] imply:

Corollary 5. *The MWS problem for (P_6, C_4)-free graphs is solvable in time $\mathcal{O}(n^3 m)$.*

Note that specific graphs are not prime.

Corollary 6. *Prime* (P_6,C_4)*-free graphs with at least* 11 *vertices and without clique cutset are nearly chordal.*

6 Combining Decomposition by Clique Cutsets and by Homogeneous Sets

In this section, we combine decomposition by clique cutsets and decomposition by homogeneous sets to a binary decomposition tree which gives a refinement of the two decompositions obtained separately.

Consider a graph $G = (V, E)$. Let $G[V_i]$ be the subgraph of G induced by $V_i \subseteq V$. We study the decomposition ϕ of a G defined as follows. If G has a clique cutset $C \subset V$, then G is decomposed into subgraphs $G_1 = G[V_1]$ and $G_2 = G[V_2]$ where $V = V_1 \cup V_2$ and $C = V_1 \cap V_2$. Whitesides [23] (see also Tarjan [22]) showed that given a decomposition of a graph G into two graphs G_1, G_2 as above, if the MWS problem for G_1, G_2 can be computed in polynomial time, then so can the problem for G.

If G has a homogeneous set H, then G is decomposed into induced subgraphs $G_1 = G[V_1]$ and $G_2 = G[V_2]$ where $V_1 = H$ and $V_2 = V - H \cup \{h\}$ for some vertex h in H. It is easy to see that given a decomposition of a graph G into two graphs G_1, G_2 as above, if the MWS problem for G_1, G_2 can be computed in polynomial time, then so can the problem for G.

We can recursively decompose G_1 and G_2 in the same way, until we obtain ϕ-prime graphs (i.e., graphs that have no clique cutset, and no homogeneous set). This decomposition can be represented by a binary tree $T(G)$ whose root is G, the two children of G are G_1 and G_2, which are in turn the roots of subtrees representing the decompositions of G_1 and G_2. Each leaf of $T(G)$ corresponds to an induced ϕ-prime subgraph of G.

Theorem 6. *For any graph* G, $T(G)$ *contains* $\mathcal{O}(n^2)$ *nodes.*

Proof. We will show that each internal node of $T(G)$ can be labeled with a distinct 2-tuple (a, b) where a, b are two vertices of G. We only need to label internal nodes that correspond to graphs with at least three vertices.

Let G_X denote the induced subgraph of G that corresponds to an internal node X of $T(G)$. If G_X is decomposed by a clique cutset C into two graphs G_1, G_2, then label X with (a, b) where a is any vertex $G_1 - C$, and b is any vertex $G_2 - C$ (we say X is a *node of type* 1). If G_X has a homogeneous set H, and G is decomposed into subgraphs $G_1 = G[V_1]$ and $G_2 = G[V_2]$ where $V_1 = H$ and $V_2 = V - H \cup \{h\}$ for some vertex h in H, then we label X with (a, b), where a is any vertex in $V - H$, and b is a vertex in $H - \{h\}$ (we say X is a *node of type* 2). Now we show that each internal node in $T(G)$ has a distinct 2-tuple.

Assume there are two nodes A, B in $T(G)$ with the same 2-tuple (x, y), in particular, we have $x, y \in G_A \cap G_B$. Suppose first that B is a descendant of A. Our choice of the labels implies that, whether A is of type 1 or 2, there is at least one vertex in the label of G_A that does not belong to G_B, a contradiction.

Now, we may assume A is not a descendant of B and B is not a descendant of A. Let X be the lowest common ancestor of A and B in $T(G)$. For simplicity, we may assume that A (B) either is the left (right) child of X, or is a descendant of the left (right) child of X. If X is a node of type 2, then A and B can have at most one vertex in common, and thus cannot have the same 2-tuple. So, X must be a node of type 1 with a clique cutset C. We thus have $x, y \in C$ implying xy is an edge. This is a contradiction since we chose x to be nonadjacent to y. □

Corollary 7. *If the MWS problem can be solved in polynomial time for every ϕ-prime subgraph of a graph G, then so can the problem for G.* □

As an example, we will study the corresponding decomposition of (P_5,P)-free graphs. As a first step, we show that these graphs are nearly perfect.

7 Prime (P_5,P)-Free Graphs and Clique Separators

In [6, 17], a (robust, see [6]) polynomial time solution for the (unweighted) Maximum Stable Set Problem on (P_5, P)-free graphs was given. The complexity of the MWS problem on (P_5, P)-free graphs remained an open question in [6, 17]. In this section, we will show that prime (P_5, P)-free graphs are nearly perfect which implies that the MWS problem is solvable in polynomial time on this graph class.

Theorem 7. *Prime (P_5, P)-free graphs are nearly perfect.*

For space limitations, we skip all remaining proofs (which are the technically most involved of this paper).

Corollary 8. *The Maximum Weight Stable Set Problem can be solved in polynomial time for (P_5, P)-free graphs.*

However, a detailed structure analysis gives a much better result:

Theorem 8. *Prime (P_5,P)-free graphs without clique cutset are nearly chordal.*

Corollary 9. *The MWS problem for (P_5, P)-free graphs is solvable in time $\mathcal{O}(n^3m)$.*

Corollary 9 follows from Corollary 7.

8 Conclusion

In this paper, we give new applications of the clique separator approach, combine it with the decomposition by homogeneous sets and improve some known polynomial time results for the unweighted Maximum Stable Set problem which were obtained by various methods such as struction, augmenting and elimination of α-redundant vertices to the weighted MWS problem.

It remains a challenging task to study under which conditions a method which works for the unweighted MS problem can be replaced by the decomposition studied in this paper. Moreover, the combined decomposition should be applied to other cases where augmenting and other methods for MS work well. In particular, it might be interesting whether the $\mathcal{O}(n^7)$ time augmenting algorithm for the MS problem on (P_7,P)-free graphs given by Alekseev and Lozin in [2] can be explained in terms of our approach.

References

1. V.E. Alekseev, On easy and hard hereditary classes of graphs with respect to the independent set problem, *Discrete Applied Math.* 132 (2004) 17-26
2. V.E. Alekseev, V.V. Lozin, Augmenting graphs for independent sets, *Discrete Applied Math.* 145 (2004) 3-10
3. J. Bang-Jensen, J. Huang, G. MacGillivray, A. Yeo, Domination in convex bipartite and convex-round graphs, manuscript 2002
4. A. Brandstädt, F.F. Dragan, On the linear and circular structure of (claw,net)-free graphs, *Discrete Applied Math.* 129 (2003) 285-303
5. A. Brandstädt, V.B. Le, and J.P. Spinrad, Graph Classes: A Survey, *SIAM Monographs on Discrete Math. Appl., Vol.* 3, SIAM, Philadelphia (1999)
6. A. Brandstädt and V.V. Lozin, A note on -redundant vertices in graphs, *Discrete Applied Math.* 108 (2001) 301-308
7. M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas, The Strong Perfect Graph Theorem, manuscript 2002
8. D.G. Corneil, H. Lerchs, and L.K. Stewart-Burlingham, Complement reducible graphs, *Discrete Applied Math.* 3 (1981) 163-174
9. D.G. Corneil, Y. Perl, and L.K. Stewart, Cographs: recognition, applications, and algorithms, *Congressus Numer.* 43 (1984) 249-258
10. D.G. Corneil, Y. Perl, and L.K. Stewart, A linear recognition algorithm for cographs, *SIAM J. Computing* 14 (1985) 926-934
11. C. Ebenegger, P.L. Hammer, and D. de Werra, Pseudo-Boolean functions and stability of graphs, *Annals Discrete Math.* 19 (1984) 83-98
12. A. Frank, Some polynomial algorithms for certain graphs and hypergraphs, *Proceedings of the Fifth British Combinatorial Conference* (Univ. Aberdeen, Aberdeen 1975) 211-226; *Congressus Numerantium* No. XV, Utilitas Math., Winnipeg, Man. (1976)
13. M.C. Golumbic, P.L. Hammer, Stability in circular-arc graphs, *J. Algorithms* 9 (1988) 314-320
14. M. Grötschel, L. Lovász, and A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica* 1 (1981) 169-197
15. P.L. Hammer, N.V.R. Mahadev, and D. de Werra, Stability in CAN-free graphs, *J. Combin. Th.* (B) 38 (1985) 23-30
16. P.L. Hammer, N.V.R. Mahadev, and D. de Werra, The struction of a graph: Application to CN-free graphs, *Combinatorica* 5 (1985) 141-147
17. V.V. Lozin, Stability in P_5- and banner-free graphs, *European J. Oper. Res.* 125 (2000) 292-297
18. R.M. McConnell and J. Spinrad, Modular decomposition and transitive orientation, *Discrete Math.* 201 (1999) 189-241

19. R.H. Möhring and F.J. Radermacher, Substitution decomposition for discrete structures and connections with combinatorial optimization, *Annals of Discrete Math.* 19 (1984) 257-356
20. R. Mosca, Stable sets in certain P_6-free graphs, *Discrete Applied Math.* 92 (1999) 177-191
21. S. Poljak, A note on stable sets and colorings of graphs, *Commun. Math. Univ. Carolinae* 15 (1974) 307-309
22. R.E. Tarjan, Decomposition by clique separators, *Discrete Math.* 55 (1985) 221-232
23. S. H. Whitesides, A method for solving certain graph recognition and optimization problems, with applications to perfect graphs, in: Berge, C. and V. Chvátal (eds), *Topics on perfect graphs*, North-Holland, Amsterdam, 1984

Smoothed Analysis of Integer Programming*

Heiko Röglin and Berthold Vöcking

Department of Computer Science – RWTH Aachen
{roeglin,voecking}@cs.rwth-aachen.de

Abstract. We present a probabilistic analysis of integer linear programs (ILPs). More specifically, we study ILPs in a so-called smoothed analysis in which it is assumed that first an adversary specifies the coefficients of an integer program and then (some of) these coefficients are randomly perturbed, e.g., using a Gaussian or a uniform distribution with small standard deviation. In this probabilistic model, we investigate structural properties of ILPs and apply them to the analysis of algorithms. For example, we prove a lower bound on the slack of the optimal solution. As a result of our analysis, we are able to specify the smoothed complexity of classes of ILPs in terms of their worst case complexity. For example, we obtain polynomial smoothed complexity for packing and covering problems with any fixed number of constraints. Previous results of this kind were restricted to the case of binary programs.

1 Introduction

Many algorithmic problems are hard with respect to worst-case instances but there are algorithms for these problem that work quite efficiently on "typical instances", that is, on instances occurring frequently in practice. Finding an adequate theoretical model for typical instances, however, is a challenging task. A reasonable approach seems to be to represent typical instances in form of a probability distribution on the set of possible inputs. A classical average-case analysis begins with the specification of the input distribution. Usually, this is just a simple uniform distribution. The dilemma with such an approach is that any fixed input distribution can be argued to be not the right, typical one. During the last years there has been an increased interest in more general input models and more robust kinds of probabilistic analyses that do not only hold for particular input distributions. An example for such a concept is the so-called *smoothed analysis* of the Simplex algorithm by Spielman and Teng [12]. They assume that first an adversary specifies the input numbers of an LP and then these adversarial numbers are slightly perturbed at random using a Gaussian distribution with specified standard deviation. Spielman and Teng show that the expected running time of the Simplex algorithm under such random perturbations is bounded polynomially in the size of the input and the reciprocal of

* Supported in part by the EU within the 6th Framework Programme under contract 001907 (DELIS).

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 276–290, 2005.

the standard deviation. Intuitively, this means that the running time function of the Simplex algorithm shows superpolynomial behavior only at some isolated peaks.

Beier and Vöcking [5] generalize smoothed analysis towards discrete optimization problems. In particular, they study optimization problems that can be represented in form of binary programs. A *linear binary optimization problem* is defined by a set of linear constraints and a linear objective function over some subset $\mathcal{S} \subseteq \{0,1\}^n$. By parametrizing which constraints are of stochastic and which are of adversarial nature, it is possible to randomize some of the constraints without destroying the combinatorial structure described by other constraints. Their analysis covers various probability distributions for the choice of the stochastic numbers and includes smoothed analysis with Gaussian and other kinds of perturbation models as a special case. It is shown that a binary optimization problem has polynomial smoothed complexity if and only if it has random pseudopolynomial complexity, i.e., the unary variant of the problem is in ZPP. Other results on the smoothed and average-case analysis of discrete optimization problems can be found, e.g., in [1, 4, 2, 3, 6, 8, 10, 11]. All these results are restricted to problems that can be written in form of a binary optimization problem.

In this paper, we extend the results of Beier and Vöcking [5] from binary towards integer linear programs (ILPs), that is, we assume that the variables have a finite domain $\mathcal{D} \subset \mathbb{Z}$ instead of just $\{0,1\}$. We investigate structural properties of ILPs and, as a result of our analysis, we are able to describe the smoothed complexity of classes of ILPs in terms of their worst case complexity. In particular, we show that any class of ILPs with polynomially bounded domain has polynomial smoothed complexity if and only if it has random pseudopolynomial complexity. For example, our characterization implies polynomial smoothed (average) complexity for packing and covering problems with any fixed number of constraints since these classes of ILPs admit pseudopolynomial time algorithms. On the other hand, packing and covering problems with an unbounded number of constraints do not have polynomial smoothed complexity, unless ZPP = NP, as these classes are strongly NP-hard[1].

Outline. In the next section, we define the considered probabilistic model and state our results in a formal way. The probabilistic analysis is presented in Section 2. It is centered around structural properties of integer linear programs, called *loser* and *feasibility gaps*. Finally, in Section 3, we show how to exploit these gaps algorithmically in form of an adaptive rounding scheme increasing the accuracy of calculation until the optimal solution is found.

1.1 Problems and Model

Our analysis deals with *integer linear programs (ILPs)*. W.l.o.g. we consider maximization programs with $\leq$-constraints of the following standard form:

[1] An NP-hard problem is called *strongly NP-hard* if it remains NP-hard even if all input numbers are encoded in unary (see e.g. [9]).

$$\max \quad c^T x \tag{1}$$
$$\text{s.t. } Ax \leq b \tag{2}$$
$$x \in \mathcal{D}^n, \tag{3}$$

where $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $c \in \mathbb{R}^n$, and $\mathcal{D} \subset \mathbb{Z}$. In our analysis, we consider *classes of ILPs*, that is, we place certain restrictions on ILPs. Packing and covering ILPs are good examples for such classes. In a *packing ILP* all coefficients are non-negative, the objective is $\max c^T x$ and all constraints are of the form $Ax \leq b$. In a *covering ILP* all coefficients are non-negative as well, the objective is $\min c^T x$ and all constraints are of the form $Ax \geq b$. Both in packing and in covering ILPs there are constraints which ensure that $x \geq 0$ holds in every feasible solution. As another example, one can also place restrictions on the number of allowed constraints. Such classes are, e.g., specified in the compendium of NP optimization problems [7]. For example, packing ILPs with only one constraint correspond to the INTEGER KNAPSACK PROBLEM, and packing ILPs with a constant number k of constraints correspond to the MAXIMUM INTEGER k-DIMENSIONAL KNAPSACK PROBLEM.

Description of the Probabilistic Input Model. Smoothed analysis assumes a semi-random input model: First, an adversary specifies all input numbers (coefficients in A and c as well as all thresholds in b), then some of the coefficients and thresholds are randomly perturbed. We assume that all numbers specified by the adversary are from the interval $[-1, 1]$. Observe that this is not a restriction as every ILP can be brought into this form by scaling the linear expressions that violate this assumption. In this extended abstract, we assume that the adversarial numbers in the constraints, i.e. the coefficients in A and the thresholds in b, are then randomly perturbed by adding an independent random number to each of them. (For an outline of alternative perturbation models see Section 1.3.) Spielman and Teng use Gaussian perturbations [12]. Following [5], we use a more general perturbation model: The random numbers that are added to the adversarial numbers are drawn according to a specified familiy of probability distributions satisfying the following conditions. Let $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be a density function such that $\sup_s(f(s)) = 1$ and $E := \int_{\mathbb{R}} |s| f(s) ds$ is finite. In words, the random variable described by f has "maximum density equal to 1" and a "finite expected absolute mean value". Function f is called the *perturbation model*. For $\phi \geq 1$, we define f_ϕ by scaling f, that is, $f_\phi(s) = \phi f(s\phi)$, for every $s \in \mathbb{R}$. This way it holds $\sup_s(f_\phi(s)) = \phi$ and $\int_{\mathbb{R}} |s| f_\phi(s) ds = E/\phi$. Now we obtain ϕ-perturbations according to perturbation model f by adding an independent random variable with density f_ϕ to each coefficient in A and each threshold in b.

For example, one obtains the Gaussian perturbation model from [12] by choosing f to be the Gaussian density with standard deviation $(2\pi)^{-1/2}$. A non-negative domain for the random numbers can be obtained, e.g., by choosing f to be the density of the uniform distribution over $[0, 1]$. In [12], the running time is described in terms of the standard deviation σ. Following [5], we describe the running time in terms of the density parameter ϕ. For the Gaussian and the

uniform distribution these two parameters are closely related; in both cases, ϕ is proportional to $1/\sigma$. Intuitively, ϕ can be seen as a measure specifying how close the probabilistic analysis is to a worst-case analysis. A worst-case instance can be interpreted as a stochastic instance in which the probability mass for each stochastic number is mapped to a single point. Thus, the larger ϕ, the closer we are to a worst-case analysis.

Definition of Smoothed Complexity. The smoothed complexity of a class of ILPs Π with an associated perturbation model f is given in terms of the input length N and the parameter ϕ. First of all, the definition of the input length needs some clarification as some of the input numbers are assumed to be random variables following continuous probability distributions. These numbers are irrational with probability 1 but we define that each of these numbers has a virtual length of one. (This way, we ensure $N \geq nk$.) The bits of the stochastic numbers can be accessed by asking an oracle in time $O(1)$ per bit. The bits after the binary point of each stochastic number are revealed one by one from left to right. As one of the results of our probabilistic analysis, we will see that $O(\log n)$ revealed bits per number are sufficient to determine the optimal solution with high probability. The deterministic part of the input[2] does not contain irrational numbers and can be encoded in an arbitrary fashion. Let I_N denote the set of possible adversarial inputs for Π of length N. For an instance $I \in I_N$, let $I + f_\phi$ denote the random instance that is obtained by a ϕ-perturbation of I. We say that Π has *polynomial smoothed complexity* under f if and only if it admits a polynomial P and an algorithm A whose running time T satisfies

$$\mathbf{Pr}\left[T(I + f_\phi) \geq P\left(N, \phi, \frac{1}{\varepsilon}\right)\right] \leq \varepsilon \ ,$$

for every $N \in \mathbb{N}$, $\phi \geq 1$, $\varepsilon \in (0, 1]$, $I \in \mathcal{I}_N$, that is, with probability at least $1 - \varepsilon$ the running time of $\mathcal{A}$ is polynomially bounded in the input length N, the perturbation parameter ϕ, and the reciprocal of ε. For a discussion of this definition see [5].

1.2 Our Results

We show that the smoothed complexity of ILPs can be characterized in terms of their worst-case complexity. For a class of ILPs Π, let Π_u denote the corresponding optimization problem in which all numbers in the constraints are assumed to be integers in unary representation instead of randomly chosen real-valued numbers. We say that the domain $\mathcal{D} \subset \mathbb{Z}$ of the decision variables is *polynomially bounded* if the cardinality of $\mathcal{D}$ can be bounded by a polynomial in the number of variables n.

Theorem 1. *A class Π of ILPs with polynomially bounded domain has polynomial smoothed complexity if and only if $\Pi_u \in ZPP$.*

[2] In this extended abstract, the deterministic part consists only of the coefficients of the objective function.

In other words, Π has polynomial smoothed complexity if it admits a (possibly randomized) algorithm with (expected) pseudopolynomial worst-case running time. If we apply this theorem to packing and covering problems then we can even drop the restriction on the domain as perturbed instances of these problems have a polynomially bounded domain with high probability.

Theorem 2. *A class Π of packing (covering) ILPs has polynomial smoothed complexity if and only if $\Pi_u \in ZPP$.*

This characterization shows that strong NP-hard classes like general packing or covering ILPs do not have polynomial smoothed complexity, unless $ZPP = NP$. On the other hand, packing and covering problems with a fixed number of constraints like, e.g., in the MAXIMUM INTEGER (k-DIMENSIONAL) KNAPSACK PROBLEM have polynomial smoothed complexity as they admit pseudopolynomial time algorithms. The same is true for ILPs with polynomially bounded domain and a fixed number of constraints. The results for packing and covering problems should only be seen as examples of our analysis. In fact, the given characterization can easily be extended to classes of ILPs in which not all constraints are required to be packing or covering constraints but only one of the constraints needs to be a packing or covering constraint.

Technical Comparison to Previous Work. In this paper we present a generalization of the smoothed analysis for binary optimization problems presented in [5] towards integer optimization problems. The rough course of the probabilistic analysis presented in the subsequent sections is similar to the analysis from [5]: We prove certain structural properties which are then exploited algorithmically in form of an adaptive rounding scheme using pseudopolynomial algorithms as a subroutine. In particular, we present a probabilistic analysis showing that it is sufficient to reveal only a logarithmic number of bits of each stochastic number in order to determine the optimal solution. We want to remark, however, that the generalization of this result from the binary to the integer case is not straightforward but technically difficult in several aspects. The major challenge we have to tackle is that the previous probabilistic analysis heavily relies on the fact that variables have only a $\{0, 1\}$ domain. For example, the previous analysis uses the existence of 0 entries in any solution (except 1^n) in order to place assumptions on subsets of solutions sharing a 0 at the same position. Observe that assumptions on the values of the solutions in such subsets do not effect the random coefficients at which all these solutions take the value 0. Obviously, this elementary trick fails already when going from a binary to a tertiary domain. In this paper, we use a different kind of analysis that places assumptions on subsets of solutions in such a way that only values of linear combinations of pairs of random coefficients are revealed. In the subsequent analysis, the knowledge about these linear combinations is taken into account carefully.

1.3 Alternative Perturbation Models

Actually, the class of perturbation models to which our analysis can be applied is far more general than the one described above. For example, in case of packing

or covering constraints one does not need to perturb the thresholds but only the coefficients. More importantly, as in the analysis of the binary case in [5] not all constraints need to be randomly perturbed. Instead, one can explicitly distinguish between those linear expressions, i.e. objective function and constraints, that shall be of adversarial and those that shall be of stochastic nature. In particular, our analysis also covers the situation that only the coefficients of the objective function or only the coefficients and the threshold of one constraint are randomly perturbed. This is important if some of the linear expressions define an underlying problem structure which should not be touched by the randomization. Furthermore, we can even drop the assumption that the expressions which are of adversarial nature are linear. This assumption is only needed for stochastic expressions. Finally, we can also extend the perturbation model in such a way that the so-called "zero structure of ILPs" is preserved, that is, coefficients set to 0 by the adversary need not to be randomly perturbed. Due to space limitations we have to shift a more detailed description of these extensions to a full version of this paper.

2 Probabilistic Analysis of ILPs

In order to prepare the proof of Theorem 1, we will analyze structural properties of semi-random ILPs. Let $I = (A, b, c)$ be an ILP with n integer variables $x_1, \ldots, x_n$ with domain $\mathcal{D}$ which has been generated according to the semi-random input model described above. We rank all solutions from $\mathcal{D}^n$ according to their objective value in non-decreasing order, i.e. we assume the objective function has to be maximized. Solutions with the same objective values are ranked in an arbitrary but fixed fashion. Throughout this analysis let $m = |\mathcal{D}|$ and $m_{\max} = \max\{|x| \,|x \in \mathcal{D}\}$ and let $[n]$ denote the set $\{1, \ldots, n\}$. Note that $m \leq 2m_{\max} + 1$ holds for every domain $\mathcal{D}$.

Loser and Feasibility Gap for a Single Constraint. At first, we will define and analyze two structural properties called "loser" and "feasibility gap" only in the case that the set of feasible solutions is described by exactly one constraint. We assume that this constraint is of the form $w^T x = w_1 x_1 + \ldots + w_n x_n \leq t$ where the coefficients $w_1, \ldots, w_n$ correspond to indepedent random variables following possibly different probability distributions with bounded densities $f_1, \ldots, f_n$, respectively. For $i \in [n]$, let $\phi_i = \sup_{s \in \mathbb{R}} f_i(s)$ and $\phi = \max_{i \in [n]} \phi_i$. For technical reasons, we have to allow further restrictions on the set of feasible solutions. To be more concrete, we assume that an arbitrary subset $\mathcal{S} \subseteq \mathcal{D}^n$ is given and that the set of feasible solutions is obtained as intersection of $\mathcal{S}$ with the half-space $\mathcal{B}$ described by the constraint $w^T x \leq t$. The *winner*, denoted by x^*, is the solution with highest rank in $\mathcal{S} \cap \mathcal{B}$. The *feasibility gap* is defined by

$$\Gamma = \begin{cases} t - w^T x^* & \text{if } \mathcal{S} \cap \mathcal{B} \neq \emptyset \\ \perp & \text{otherwise} \end{cases}.$$

In words, Γ corresponds to the slack of the winner with respect to the threshold t. A solution from $\mathcal{S}$ is called a *loser* if it has a higher rank than x^*, that is, the

losers are those solutions from $\mathcal{S}$ that are better than the winner (w.r.t. the ranking) but that are cut off by the constraint $w^T x \leq t$. The set of losers is denoted by $\mathcal{L}$. If there is no winner, as there is no feasible solution, then we define $\mathcal{L} = \mathcal{S}$. The *loser gap* is defined by

$$\Lambda = \begin{cases} \min\{w^T x - t | x \in \mathcal{L}\} & \text{if } \mathcal{L} \neq \emptyset \\ \bot & \text{otherwise} \end{cases}.$$

Our goal is to show that both the loser and the feasibility gap of a semi-random ILP are lower bounded by a polynomial in $(nm_{\max}\phi)^{-1}$ with probability close to 1.

Observe that the solution 0^n is different from all other solutions in $\mathcal{S}$ as its feasibilty does not depend on the outcome of the random coefficients $w_1, \ldots, w_n$. Suppose $0^n \in \mathcal{S}$ and 0^n has the highest rank among all solutions in $\mathcal{S}$. Then one can enforce $\Gamma = 0$ by setting $t = 0$. Similary, one can enforce $\Lambda \to 0$ for $t < 0$ and $t \to 0$. For this reason, we need to exclude the solution 0^n from our analysis. Later we will describe how the random perturbation of the threshold helps us to cope with this problem.

The key result of this section is the following lemma about the sizes of loser and feasibility gap.

Lemma 3. *Let $\mathcal{S}$ with $0^n \notin \mathcal{S}$ be choosen arbitrarily and let $c = \max_{i\in[n]} \mathbf{E}[|w_i|]$. For all ε with $\varepsilon \leq (32n^5m^7m_{\max}\phi^2)^{-1}$, $\mathbf{Pr}[\Gamma \leq \varepsilon] \leq 2(\varepsilon \cdot 32cn^5m^7m_{\max}\phi^2)^{1/3}$ and $\mathbf{Pr}[\Lambda \leq \varepsilon] \leq 2(\varepsilon \cdot 32cn^5m^7m_{\max}\phi^2)^{1/3}$.*

The proof of this lemma is subdivided into a few steps. At first, we will assume that the densities $f_1, \ldots, f_n$ have a bounded support, i.e. we assume the existence of a constant $s \in \mathbb{R}_{\geq 0}$ such that $f_i(x) = 0$ holds for every $i \in [n]$ and for every $x \notin [-s, s]$. In addition to that, we assume that the set $\mathcal{S}$ does only contain elements which are pairwise linearly independent, i.e. we assume that there do not exist two solutions $x, y \in \mathcal{S}$ such that $x = \alpha y$ or $y = \alpha x$ holds for some $\alpha \in \mathbb{R}$. In this case we can show an upper bound on the probability that the loser gap does not exceed ε. Then, we will use symmetry properties between the two gaps in order to show that bounds for the loser gap also hold for the feasibility gap and vice versa. Thus, the bound proven for the loser gap holds for the feasibility gap as well. The assumption that the set $\mathcal{S}$ does not contain linearly dependent solutions can be dropped at the cost of an extra factor m for the feasibility gap. Due to the symmetry this bound also applies to the loser gap. In the last step we will drop the assumption that the support of the densities is bounded.

Lemma 4. *Let $\mathcal{S}$ with $0^n \notin \mathcal{S}$ be choosen arbitrarily such that $\mathcal{S}$ does not contain two linearly dependent solutions. Assume $f_i(x) = 0$, for $i \in [n]$ and $x \notin [-s, s]$. Then, for all $\varepsilon \geq 0$ and for all $p \geq 1$,*

$$\mathbf{Pr}[\Lambda \leq \varepsilon] \leq \frac{1}{2p} + \varepsilon \cdot 4n^4m^6m_{\max}\phi^2 sp.$$

Proof. The role of the parameter p needs some explanation. We will show that the density of the loser gap is upper bounded by $4n^4m^6m_{\max}\phi^2sp$ if some failure event $\mathcal{E}(p)$ does not occur. It holds $\mathbf{Pr}[\mathcal{E}(p)] \leq 1/(2p)$. Thus, the first addend corresponds to the case of failure $\mathcal{E}(p)$ and the second one corresponds to the case $\neg\mathcal{E}(p)$. Note that an upper bound α on the density of Λ implies an upper bound on the probability that Λ takes a value less than or equal to ε of $\varepsilon\alpha$ since Λ takes only non-negative values.

Now we will present our approach to bound the density f_Λ of the random variable Λ. We will see that this approach fails under certain circumstances and define the failure event $\mathcal{E} := \mathcal{E}(p)$ accordingly. For each combination of $i, j \in [n]$ with $i < j$ and of $\overline{m} = (m_1, m_2, m_3, m_4) \in \mathcal{D}^4$ with linearly independent vectors (m_1, m_2) and (m_3, m_4), we define a random variable $\Lambda_{i,j}^{\overline{m}}$ in such a way that there are always indices i, j and a vector $\overline{m}$ such that $\Lambda = \Lambda_{i,j}^{\overline{m}}$ holds. Thus, it holds

$$\mathbf{Pr}[\Lambda \leq \varepsilon] = \mathbf{Pr}[\Lambda \in [0, \varepsilon]] \leq \sum_{i,j,\overline{m}} \mathbf{Pr}\left[\Lambda_{i,j}^{\overline{m}} \in [0, \varepsilon]\right].$$

For this reason, a bound on the densities of the random variables $\Lambda_{i,j}^{\overline{m}}$ implies a bound on the probability that Λ does not exceed ε.

Let x^* denote the winner, let $x^{\min}$ denote the *minimal loser*, i.e. $x^{\min} = \operatorname{argmin}\{w^T x | x \in \mathcal{L}\}$, and fix some $i, j \in [n]$ with $i < j$ and a vector $\overline{m} \in \mathcal{D}^4$ with linearly independent subvectors (m_1, m_2) and (m_3, m_4). First of all, we will formally define the random variable $\Lambda_{i,j}^{\overline{m}}$. Therefore, let $x_{i,j}^{*,m_3,m_4}$ denote the winner of those solutions x with $x_i = m_3$ and $x_j = m_4$, i.e. $x_{i,j}^{*,m_3,m_4}$ denotes the highest ranked solution in $\{x \in \mathcal{S} | x_i = m_3, x_j = m_4\} \cap \mathcal{B}$. Based on this definition, we define a set of losers

$$\mathcal{L}_{i,j}^{\overline{m}} = \{x \in \mathcal{S} \mid x_i = m_1, x_j = m_2,\ x \text{ is ranked higher than } x_{i,j}^{*,m_3,m_4}\}.$$

The minimal loser $x_{i,j}^{\min,\overline{m}}$ is defined to be the solution from $\mathcal{L}_{i,j}^{\overline{m}}$ with the smallest weight, i.e. $x_{i,j}^{\min,\overline{m}} = \operatorname{argmin}\{w^T x \mid x \in \mathcal{L}_{i,j}^{\overline{m}}\}$. Now the random variable $\Lambda_{i,j}^{\overline{m}}$ is defined to be the slack of the minimal loser $x_{i,j}^{\min,\overline{m}}$ w.r.t. the threshold t, i.e. $\Lambda_{i,j}^{\overline{m}} = w^T x_{i,j}^{\min,\overline{m}} - t$. If $\mathcal{L}_{i,j}^{\overline{m}} = \emptyset$ then $x_{i,j}^{\min,\overline{m}}$ and $\Lambda_{i,j}^{\overline{m}}$ are undefined.

One can easily argue that the requirement that Λ always takes a value equal to one of the values of the $\Lambda_{i,j}^{\overline{m}}$ is fulfilled: The winner x^* and the minimal loser $x^{\min}$ are linearly independent since they are both elements from $\mathcal{S}$. Thus, there can always be found two indices $i, j \in [n]$ with $i < j$ such that the vectors (x_i^*, x_j^*) and $(x_i^{\min}, x_j^{\min})$ are linearly independent. Setting $(m_3, m_4) = (x_i^*, x_j^*)$ and $(m_1, m_2) = (x_i^{\min}, x_j^{\min})$ yields $\Lambda = \Lambda_{i,j}^{\overline{m}}$.

In order to obtain an upper bound on the density of the random variable $\Lambda_{i,j}^{\overline{m}}$, we reduce the degree of randomness. That is, we assume the coefficients w_k with $k \neq i$ and $k \neq j$ and the sum $m_3 w_i + m_4 w_j$ to be fixed arbitrarily. An upper bound for the density of $\Lambda_{i,j}^{\overline{m}}$ holding for all deterministic choices of these random variables obviously holds for all random choices as well. The winner $x_{i,j}^{*,m_3,m_4}$ can be determined without knowing the outcome of w_i and w_j as the weights of all solutions in $\{x \in \mathcal{S} | x_i = m_3, x_j = m_4\}$ are known. Thus,

also $\mathcal{L}_{i,j}^{\overline{m}}$ is known. Since the random variables w_i and w_j affect the weight of all solutions in $\mathcal{L}_{i,j}^{\overline{m}}$ in the same fashion, also the minimal loser $x_{i,j}^{\min,\overline{m}}$ does not depend on the outcome of w_i and w_j. Hence, if the outcome of w_k with $k \neq i$ and $k \neq j$ and the sum $m_3 w_i + m_4 w_j$ are known, the loser gap $\Lambda_{i,j}^{\overline{m}}$ can be rewritten as

$$\Lambda_{i,j}^{\overline{m}} = w^T x_{i,j}^{\min,\overline{m}} - t = \kappa + m_1 w_i + m_2 w_j,$$

where κ denotes a constant depending on the fixed values of w_k with $k \neq i$ and $k \neq j$ and $m_3 w_i + m_4 w_j$. Thus, under our assumption, $\Lambda_{i,j}^{\overline{m}}$ and$m_1 w_i + m_2 w_j$ are random variables which differ only by a constant addend. In particular, upper bounds on the density of the random variable $m_1 w_i + m_2 w_j$ hold for the density of $\Lambda_{i,j}^{\overline{m}}$ as well. Recall that we still assume the sum $m_3 w_i + m_4 w_j$ to be fixed to an arbitrary value $z \in \mathbb{R}$. Therefore, we will determine the conditional density $g_{i,j}^{\overline{m},z}$ of $m_1 w_i + m_2 w_j$ under the condition $m_3 w_i + m_4 w_j = z$.

Let $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ denote the joint density of the random variables $A := m_1 w_i + m_2 w_j$ and $B := m_3 w_i + m_4 w_j$. Since the vectors (m_1, m_2) and (m_3, m_4) are assumed to be linearly independent, the transformation $\Phi : \mathbb{R}^2 \to \mathbb{R}^2$ with $\Phi(x, y) = (m_1 x + m_2 y, m_3 x + m_4 y)$ is bijective and can be inverted as follows

$$\Phi^{-1}(a, b) = \left(\frac{m_4 a - m_2 b}{m_1 m_4 - m_2 m_3}, \frac{m_1 b - m_3 a}{m_1 m_4 - m_2 m_3} \right).$$

In order to determine the conditional density $g_{i,j}^{\overline{m},z}$, we have to determine the Jacobian matrix M of the transformation Φ^{-1} containing the partial derivations of Φ^{-1} as matrix entries. With $d = m_1 m_4 - m_2 m_3$ it holds

$$M = \begin{pmatrix} \frac{m_4}{d} & -\frac{m_2}{d} \\ -\frac{m_3}{d} & \frac{m_1}{d} \end{pmatrix}.$$

The determinant of the Jacobian matrix is $1/d$. Due to the independence of the random variables w_i and w_j, the joint density f of A and B can be written as

$$\begin{aligned} f(a, b) &= |\det M| \cdot f_i(\Phi_1^{-1}(a, b)) \cdot f_j(\Phi_2^{-1}(a, b)) \\ &= \frac{1}{|d|} \cdot f_i \left(\frac{m_4 a - m_2 b}{d} \right) \cdot f_j \left(\frac{m_1 b - m_3 a}{d} \right) \\ &\leq \frac{\phi^2}{|d|} \leq \phi^2. \end{aligned}$$

The conditional density $g_{i,j}^{\overline{m},z}$ can be expressed as follows

$$g_{i,j}^{\overline{m},z}(x) = \frac{f(x, z)}{\int_{\mathbb{R}} f(x, z)\, dx} = \frac{f(x, z)}{f_{m_3 w_i + m_4 w_j}(z)},$$

where $f_{m_3w_i+m_4w_j}$ denotes the density of the random variable $B = m_3w_i+m_4w_j$. Thus, for all $x \in \mathbb{R}$, it holds

$$g_{i,j}^{\overline{m},z}(x) \leq \frac{\phi^2}{f_{m_3w_i+m_4w_j}(z)}, \tag{4}$$

where $f_{m_3w_i+m_4w_j}$ denotes the density of the random variable $m_3w_i + m_4w_j$.

Hence, $g_{i,j}^{\overline{m},z}$ cannot be upper bounded since, in general, the denominator in (4) can become arbitrarily small. Therefore, we have to restrict the possible choices for z to the set $\mathbb{R}\backslash M_{i,j}^{m_3,m_4}$ with

$$M_{i,j}^{m_3,m_4} = \left\{ z \in \mathbb{R} \,\middle|\, 0 \leq f_{m_3w_i+m_4w_j}(z) \leq \frac{1}{4n^2m^2m_{\max}sp} \right\}.$$

We will denote the event that $m_3w_i + m_4w_j$ takes a value from $M_{i,j}^{m_3,m_4}$ by $\mathcal{E}_{i,j}^{m_3,m_4}$. In case of $\neg\mathcal{E}_{i,j}^{m_3,m_4}$ the conditional density $g_{i,j}^{\overline{m},z}$ is bounded from above by $4n^2m^2m_{\max}sp\phi^2$. Hence, $\mathbf{Pr}\left[\Lambda_{i,j}^{\overline{m}} \in [0,\varepsilon] | \neg\mathcal{E}_{i,j}^{m_3,m_4}\right] \leq \varepsilon \cdot 4n^2m^2m_{\max}\phi^2sp$. Due to the bounded support of the densities f_i and f_j, it is not very likely that the event $\mathcal{E}_{i,j}^{m_3,m_4}$ occurs. Let $\mathcal{E}$ denote the event that, for at least one combination of $i, j \in [n]$ and $m_3, m_4 \in \mathcal{D}$, the event $\mathcal{E}_{i,j}^{m_3,m_4}$ occurs, that is, $\mathcal{E}$ denotes the union of all these events. An easy (though long) calculation shows

$$\mathbf{Pr}\left[\Lambda \in [0,\varepsilon] | \neg\mathcal{E}\right] \leq \varepsilon \cdot 4n^4m^6m_{\max}\phi^2sp \tag{5}$$

and

$$\mathbf{Pr}\left[\mathcal{E}\right] \leq \frac{1}{2p} \tag{6}$$

and, therefore,

$$\mathbf{Pr}\left[\Lambda \leq \varepsilon\right] \leq \mathbf{Pr}\left[\mathcal{E}\right] + \mathbf{Pr}\left[\Lambda \in [0,\varepsilon] | \neg\mathcal{E}\right] \leq \frac{1}{2p} + \varepsilon \cdot 4n^4m^6m_{\max}\phi^2sp.$$

□

Now we will show that Lemma 4 holds for the feasibility gap as well. First of all, we have to generalize the definitions of loser and feasibility gap a little bit. Let $\Lambda(t)$ denote the loser gap w.r.t. the constraint $w^Tx \leq t$ and let $\Gamma(t)$ denote the feasibility gap w.r.t. to this constraint.

Lemma 5. *Let $t \in \mathbb{R}$ and $\varepsilon \in \mathbb{R}_{\geq 0}$ arbitrary then* $\mathbf{Pr}\left[\Lambda(t) < \varepsilon | \neg\mathcal{E}\right] = \mathbf{Pr}\left[\Gamma(t+\varepsilon) < \varepsilon | \neg\mathcal{E}\right]$.

This lemma can be proven by arguments similar to those used in the proof of Lemma 9 in [5].

Next, we will drop the assumption that the set of feasible solutions $\mathcal{S}$ does not contain linearly dependent solutions and obtain the following result.

Lemma 6. *Let $\mathcal{S}$ with $0^n \notin \mathcal{S}$ be choosen arbitrarily. Assume $f_i(x) = 0$, for $i \in [n]$ and $x \notin [-s,s]$. Then, for all $\varepsilon \geq 0$ and for all $p \geq 1$,* $\mathbf{Pr}\left[\Gamma < \varepsilon | \neg\mathcal{E}\right] \leq \varepsilon \cdot 4n^4m^7m_{\max}\phi^2sp$ *and* $\mathbf{Pr}\left[\Lambda < \varepsilon | \neg\mathcal{E}\right] \leq \varepsilon \cdot 4n^4m^7m_{\max}\phi^2sp$.

Proof. The main idea of the proof is to partition the set $\mathcal{S}$ into m classes $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(m)}$ such that none of these classes contains two solutions which are linearly dependent. Let $\mathcal{D} = \{d_1, \ldots, d_m\}$. If $0 \notin \mathcal{D}$, such a partition can simply be created by setting $\mathcal{S}^{(k)} = \{x \in \mathcal{S} | x_1 = d_k\}$, for $k \in [m]$. Otherwise, we assume w.l.o.g. $d_m = 0$ and we set, for $k \in [m-1]$,

$$\mathcal{S}^{(k)} = \{x \in \mathcal{S} \mid \exists i \in [n] : x_1 = \ldots = x_{i-1} = 0 \text{ and } x_i = d_k\}.$$

For each of these classes a feasibility gap $\Gamma^{(k)}$ is defined. First we define the winner $x^{*,(k)}$ w.r.t. $\mathcal{S}^{(k)}$ to be that element from $\mathcal{S}^{(k)} \cap \mathcal{B}$ which is ranked highest. The feasibility gap $\Gamma^{(k)}$ is simply defined as $t - w^T x^{*,(k)}$, if $\mathcal{S}^{(k)} \neq \emptyset$, and $\perp$ otherwise. Since the winner x^* of the original problem is contained in one of the classes $\mathcal{S}^{(k)}$, the feasibility gap Γ always takes the value of one of the variables $\Gamma^{(k)}$. Observe that Lemma 4 can be applied to the subproblems defined by the classes $\mathcal{S}^{(k)}$ since these classes do not contain linearly dependent solutions. Hence, we can combine equation (5) and Lemma 5 to obtain $\mathbf{Pr}\left[\Gamma^{(k)} \leq \varepsilon | \neg\mathcal{E}\right] \leq \varepsilon \cdot 4n^4 m^6 m_{\max} \phi^2 sp$. Thus, it holds

$$\mathbf{Pr}\left[\Gamma \leq \varepsilon | \neg\mathcal{E}\right] \leq \sum_{k=1}^{m} \mathbf{Pr}\left[\Gamma^{(k)} \leq \varepsilon | \neg\mathcal{E}\right] \leq \varepsilon \cdot 4n^4 m^7 m_{\max} \phi^2 sp.$$

The result on the loser gap follows by another application of Lemma 5. □

Now we will drop the assumption that the densities $f_1, \ldots, f_n$ have bounded supports and finish the proof of Lemma 3.

Proof (Lemma 3). The main idea is to choose some constant $s \in \mathbb{R}$ such that the probability that one of the coefficients $w_1, \ldots, w_n$ takes a value outside of the interval $[-s, s]$ is bounded above by $1/(2p)$. We set $s = 2npc$. For $i \in [n]$, let $\mathcal{G}_i$ denote the event that $w_i \notin [-s, s]$ and let $\mathcal{G}$ denote the union of these events. An application of Markov's inequality shows $\mathbf{Pr}[\mathcal{G}] \leq 1/(2p)$. For the conditional density functions it holds

$$f_{i|\neg\mathcal{G}}(x) = \begin{cases} 0 & \text{if } x \notin [-s, s] \\ \frac{f_i(x)}{\mathbf{Pr}[w_i \in [-s,s]]} & \text{otherwise} \end{cases} \leq \begin{cases} 0 & \text{if } x \notin [-s, s] \\ 2f_i(x) & \text{otherwise} \end{cases}.$$

Thus, the densities of the random variables $w_1, \ldots, w_n$ have a bounded support under the condition $\neg\mathcal{G}$. We define $\mathcal{F} = \mathcal{E} \cup \mathcal{G}$ to be the failure event. We can bound the probability that the loser gap or the feasibility gap does not exceed ε under the condition $\neg\mathcal{F}$. We have seen, that the condition $\neg\mathcal{G}$ leads to a conditional density which is by a factor of at most 2 larger than the unconditional density Hence, Lemma 6 yields $\mathbf{Pr}[\Lambda < \varepsilon | \neg\mathcal{F}] \leq \varepsilon \cdot 32cn^5 m^7 m_{\max} \phi^2 p^2$. Furthemore, it holds $\mathbf{Pr}[\mathcal{F}] = \mathbf{Pr}[\mathcal{E} \cup \mathcal{G}] \leq \mathbf{Pr}[\mathcal{E}] + \mathbf{Pr}[\mathcal{G}] \leq \frac{1}{p}$. Thus, we obtain

$$\mathbf{Pr}[\Lambda < \varepsilon] \leq \frac{1}{p} + \varepsilon \cdot 32cn^5 m^7 m_{\max} \phi^2 p^2.$$

Setting $p = (\varepsilon \cdot 32cn^5 m^7 m_{\max} \phi^2)^{-1/3}$ yields the desired result. The upper bound on ε is due to the assumption $p \geq 1$. The claim about the feasibility gap follows analogously. □

Loser and Feasibility Gap for Multiple Constraints. Assume there are $k \geq 2$ constraints. W.l.o.g. these constraints are of the form $Ax \leq b$ with $A \in \mathbb{R}^{k \times n}$ and $b \in \mathbb{R}^k$, and the set of points satisfying these constraints are $\mathcal{B}_1, \ldots, \mathcal{B}_k$, respectively. We generalize the definition of feasibility and loser gap as follows. Given a set of solutions $\mathcal{S} \subseteq \mathcal{D}^n$ and a ranking, the winner x^* is the highest ranked solution in $\mathcal{S} \cap \mathcal{B}_1, \ldots, \cap \mathcal{B}_k$. The feasibility gap for multiple constraints is the minimal slack of x^* over all constraints, that is, $\Gamma = \min_{j \in [k]}\{(b - Ax)_j\}$, if x^* exists, and $\Gamma = \perp$ otherwise. The set of losers $\mathcal{L}$ consists of all solutions from $\mathcal{S}$ that have a higher rank than x^*. We define the loser gap for multiple constraints by $\Lambda = \min_{x \in \mathcal{L}} \max_{j \in [k]}\{(Ax - b)_j\}$, if $\mathcal{L} \neq \emptyset$, and $\Lambda = \perp$, otherwise. (See [5] for a motivation of this definition.)

Lemma 7. *Let* $c = \max_{j \in [k]} \max_{i \in [n]} \mathbf{E}\left[|A_{j,i}|\right]$ *and let* k *denote the number of constraints. Then, for all* ε *with* $\varepsilon \leq (32n^5m^7m_{\max}\phi^2)^{-1}$, $\mathbf{Pr}\left[\Gamma \leq \varepsilon\right] \leq 2k(\varepsilon \cdot 32cn^5m^7m_{\max}\phi^2)^{1/3}$ *and* $\mathbf{Pr}\left[\Lambda \leq \varepsilon\right] \leq 2k(\varepsilon \cdot 32cn^5m^7m_{\max}\phi^2)^{1/3}$.

The proof of this lemma is completely analogous to the generalization to multiple constraints in the binary case in [5].

3 From Structural Properties to Algorithms

At first, we prove that a randomized pseudopolynomial algorithm implies polynomial smoothed complexity. We design an algorithm with polynomial smoothed complexity calling the pseudopolynomial algorithm with higher and higher precision until the optimal solution is found. Due to space limitations, we only present the core of the algorithm and its analysis, namely we present how to compute a certified winner when only a bounded number of bits per input number is available. The algorithm has available d bits after the binary point of each random coefficient and either outputs the true winner or, if it cannot compute such a winner as it needs more bits, it reports a failure.

Certifier. Let I denote an ILP created with the semi-random input model and let k denote the number of constraints, that is the constraints have the form $Ax \leq b$ with $A \in \mathbb{R}^{k \times n}$ and $b \in \mathbb{R}^k$. First of all, the certifier checks whether there exists an index $i \in [k]$ such that $b_i \in [-(nm_{\max} + 1)2^{-d}, 0)$ or not. In the former case, the certifier cannot compute the true winner with the given number of revealed bits d per coefficient. Otherwise, the pseudopolynomial algorithm is called to calculate the winner x' w.r.t. to the coefficients $A' := \lfloor A \rfloor_d$ and the thresholds $b' := \lfloor b \rfloor_d + (nm_{\max} + 1)2^{-d}$, where $\lfloor . \rfloor_d$ denotes the matrix or the vector that is obtained by rounding down each entry to the next multiple of 2^{-d}.

First, we will show that solutions which are feasible w.r.t. to the constraints $Ax \leq b$ stay feasible w.r.t. the constraints $A'x \leq b'$. Assume $Ax \leq b$. Since the rounding changes each coefficient by at most 2^{-d} and since, for $i \in [n]$, it holds $|x_i| \leq m_{\max}$, the j-th weight of the solution x, i.e. $(Ax)_j := a_{j,1}x_1 + \ldots + a_{j,n}x_n$, for $j \in [k]$, is changed by at most $nm_{\max}2^{-d}$. Hence,

$$A'x = \lfloor A \rfloor_b x \leq Ax + nm_{\max}2^{-d} \leq b + nm_{\max}2^{-d} \leq \lfloor b \rfloor_d + (nm_{\max} + 1)2^{-d},$$

where $\leq$ means $\leq$ in every component and, for a matrix A and a real number z, $A + z$ denotes the matrix obtained from A by adding z to each entry.

Now we must check, whether the solution x' is feasible w.r.t $Ax \leq b$ or has become feasible due to the rounding. Therefore, the certifier tests if $\lfloor A \rfloor_d x' \leq \lfloor b \rfloor_d - (nm_{\max} + 1)2^{-d}$ holds. Only in the affirmative case, the solution x' can be certified to be feasible w.r.t. the constraints $Ax \leq b$. Otherwise, the certifier cannot calculate a certified winner. Assume x' is not feasible w.r.t. to $Ax \leq b$, then, for at least one $j \in [k]$, it holds $(Ax')_j > b_j$. Hence,

$$(\lfloor A \rfloor_b x)_j \geq (Ax)_j - (nm_{\max})2^{-d} > b_j - (nm_{\max})2^{-d} \geq \lfloor b_j \rfloor_d - (nm_{\max} + 1)2^{-d}.$$

Altogether the certifier fails if, for at least one $i \in [k]$, $b_i \in [-(nm_{\max} + 1)2^{-d}, 0)$ holds or if $\lfloor A \rfloor_d x' \leq \lfloor b \rfloor_d - (nm_{\max} + 1)2^{-d}$ does not hold. Since the thresholds are random variables whose densities are bounded by ϕ, the probability of the first event is bounded from above by $k(nm_{\max} + 1)2^{-d}\phi$. In order to bound the probability of the second event, we have to distinguish between the cases that x' is feasible w.r.t. $Ax \leq b$ or not. In the former case the feasibility gap cannot exceed $(nm_{\max} + 1)2^{-d+1}$, in the latter case the loser gap cannot exceed $(nm_{\max} + 1)2^{-d+1}$.

We will further analyze the case that $\lfloor A \rfloor_d x' \leq \lfloor b \rfloor_d - (nm_{\max} + 1)2^{-d}$ does not hold. Let $j \in [k]$ with $(\lfloor A \rfloor_d x')_j > \lfloor b_j \rfloor_d - (nm_{\max} + 1)2^{-d}$. Assume that x' is feasible w.r.t the constraints $Ax \leq b$, that is x' is the true winner. Then, it holds

$$\begin{aligned}(Ax)_j &\geq (\lfloor A \rfloor_d x')_j - nm_{\max}2^{-d} \\ &> \lfloor b_j \rfloor_d - (nm_{\max} + 1)2^{-d} - nm_{\max}2^{-d} \\ &\geq b_j - (nm_{\max} + 1)2^{-d+1}.\end{aligned}$$

Thus, in this case, the feasibiliy gap cannot be larger than $(nm_{\max} + 1)2^{-d+1}$.

Now assume that x' is not feasible w.r.t the constraints $Ax \leq b$, that is x' has become feasible due to the rounding. Assume further that the loser gap is larger than $(nm_{\max} + 1)2^{-d+1}$, that is, it exists at least one $j \in [k]$ such that $(Ax')_j > b_j + (nm_{\max} + 1)2^{-d+1}$ holds. Then

$$\begin{aligned}(\lfloor A \rfloor_d x')_j &\geq (Ax')_j - nm_{\max}2^{-d} \\ &> b_j + (nm_{\max} + 1)2^{-d+1} - nm_{\max}2^{-d} \\ &\geq \lfloor b_j \rfloor_d + (nm_{\max} + 1)2^{-d+1} - nm_{\max}2^{-d} - 2^{-d} \\ &= \lfloor b_j \rfloor_d + (nm_{\max} + 1)2^{-d}.\end{aligned}$$

Thus, in contradiction to the assumption, x' is not feasible w.r.t. $A'x \leq b'$. Hence, the loser gap cannot be larger than $(nm_{\max} + 1)2^{-d+1}$ if x' has become feasible due to the rounding.

Until now, we have not yet considered the case $0^n \in \mathcal{S}$ explicitly. Assume that there exists at least one $j \in [k]$ such that $b_j < -(nm_{\max} + 1)2^{-d}$. Then 0^n is neither feasible w.r.t. $Ax \leq b$ nor w.r.t. $A'x \leq b'$. Hence, in this case, the solution 0^n does not affect our analysis. Since the certifier fails if, for at

least one $j \in [k]$, $b_j \in [-(nm_{\max}+1)2^{-d}, 0)$, also in this case, the solution 0^n does not affect the certifier. Now assume, for all $j \in [k]$, $b_j \geq 0$. Then 0^n is feasible with w.r.t. $Ax \leq b$ and $A'x \leq b'$. If 0^n is not the optimal solution w.r.t. $Ax \leq b$, then 0^n does not affect the certifier. Hence, the only case which needs to be considered in more detail is the case that 0^n is the optimal solution w.r.t. $Ax \leq b$. Oberserve that the feasibility of the solution 0^n can be verified easily. Therefore, no problem occurs in the case that 0^n is the optimal solution w.r.t. $A'x \leq b'$.

The only case which is a little bit tricky to handle is the case that 0^n is the optimal solution w.r.t. $Ax \leq b$ but that $x' \neq 0^n$ is the optimal solution w.r.t. $A'x \leq b'$. In this case, x' is rejected by the certifier since $\lfloor A \rfloor_d x' \leq \lfloor b \rfloor_d - (nm_{\max}+1)2^{-d}$ does not hold. We have to bound the probability that this case occurs. Analogous to the case $0^n \notin \mathcal{S}$, one can argue that this can only happen if the size of the loser gap Λ does not exceed $(nm_{\max}+1)2^{-d+1}$. Unfortunately, we cannot apply Lemma 7 directly since we analyzed the gaps only in the case $0^n \notin \mathcal{S}$. Instead, we exclude 0^n from the set of feasible solutions, that is we define $\mathcal{S}' = \mathcal{S} \backslash \{0^n\}$ and argue with the help of the loser gap Λ' w.r.t. to $\mathcal{S}'$. The crucial observation is that adding 0^n to the set of solutions can, in the case $b \geq 0$, only result in an increase of the size of the loser gap. The reason therefore is that, in the case $b \geq 0$, 0^n is a feasible solution which means that by adding 0^n to the set of solutions one cannot enlarge the set of losers $\mathcal{L}$. Hence, it holds $\Lambda \geq \Lambda'$ and we can make use of Lemma 7 in order to bound the probability that Λ' does not exceed $(nm_{\max}+1)2^{-d+1}$.

Adaptive Rounding. Now let us briefly sketch the missing details of the algorithm and its analysis. Until now we did not specify how the optimal solution for the rounded coefficients is actually computed. For this purpose, we use the pseudopolynomial algorithm. First, we set $d = 1$, that is we reveal only the first bit after the binary point of each coefficient. The pseudopolynomial algorithm is called to calculate the optimum w.r.t. to the rounded coefficients. If the certifier fails, the number of revealed bits d is increased by one and the pseudopolynomial algorithm and the certifier are called again. This is repeated until a certified winner can be calculated. The optimal solution is found when $d = O(\log(\phi nkm_{\max}))$, with high probability (whp). Hence, the pseudopolynomial algorithm has to deal with numbers described by $O(\log(\phi nkm_{\max}))$ bits so that its running time is bounded by $2^{O(\log(\phi nkm_{\max}))} = \text{poly}(\phi nkm_{\max})$, whp. (More details can be found in a full version of this paper.)

From Polynomial Smoothed Complexity to Pseudopolynomial Running Time. Finally, we need to show that polynomial smoothed complexity implies the existence of a randomized pseudopolynomial algorithm. This can be shown analogously to the binary case analyzed in [5].

4 Conclusions

Our probabilistic analysis shows that important classes of ILPs with a fixed number of constraints have polynomial smoothed complexity. This means that

random or randomly perturbed instances of such ILPs can be solved in polynomial time. The presented algorithmic framework giving these results uses algorithms with pseudopolynomial worst-case complexity as subroutines. Usually these pseudopolynomial time algorithms are based on dynamic programming. We want to remark that we do not believe that this approach is the most practical one to tackle ILPs of this kind. We expect that *branch and bound* and *branch and cut* heuristics are much faster than algorithms based on dynamic programming. The next challenging task is a smoothed analysis of these heuristics in order to theoretically explain their great success on practical applications. We think that the main contribution of this paper is to point out chances and limitations for such a probabilistic analysis.

References

1. C. Banderier, R. Beier, and K. Mehlhorn. Smoothed Analysis of Three Combinatorial Problems. In *Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS-2003)*, volume 97, pages 198–207, 2003.
2. R. Beier and B. Vöcking. An Experimental Study of Random Knapsack Problems. In *Proc. of the 12th Annual European Symposium on Algorithms (ESA-2004)*, pages 616–627, 2004.
3. R. Beier and B. Vöcking. Probabilistic Analysis of Knapsack Core Algorithms. In *Proc. of the 15th Annual Symposium on Discrete Algorithms (SODA-2004)*, pages 468–477, New Orleans, USA, 2004.
4. R. Beier and B. Vöcking. Random Knapsack in Expected Polynomial Time. In *Journal of Computer and System Sciences*, volume 69(3), pages 306–329, 2004.
5. R. Beier and B. Vöcking. Typical Properties of Winners and Losers in Discrete Optimization. In *Proc. of the 36th Annual ACM Symposium on Theory of Computing (STOC-2004)*, pages 343–352, 2004.
6. K. H. Borgwardt and J. Brzank. Average Saving Effects in Enumerative Methods for Solving Knapsack Problems. In *Journal of Complexity*, volume 10, pages 129–141, 1994.
7. P. Crescenzi, V. Kann, M. Halldorsson, M. Karpinski, and G. Woeginger. A compendium of NP optimization problems. http://www.nada.kth.se/~viggo/problemlist/compendium.html.
8. M. E. Dyer and A. M. Frieze. Probabilistic Analysis of the Multidimensional Knapsack Problem. In *Mathematics of Operations Research*, volume 14(1), pages 162–176, 1989.
9. M. Garey and D. Johnson. *Computers and Intractability.* Freeman, 1979.
10. A. Goldberg and A. Marchetti-Spaccamela. On Finding the Exact Solution to a Zero-One Knapsack Problem. In *Proc. of the 16th Annual ACM Symposium on Theory of Computing (STOC-1984)*, pages 359–368, 1984.
11. G. S. Lueker. Average-Case Analysis of Off-Line and On-Line Knapsack Problems. In *Journal of Algorithms*, volume 19, pages 277–305, 1998.
12. D. A. Spielman and S.-H. Teng. Smoothed Analysis of Algorithms: Why The Simplex Algorithm Usually Takes Polynomial Time. In *Journal of the ACM*, volume 51(3), pages 385–463, 2004.

Circular Ones Matrices and the Stable Set Polytope of Quasi-line Graphs

Friedrich Eisenbrand[1], Gianpaolo Oriolo[2], Gautier Stauffer[3], and Paolo Ventura[4]

[1] Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
eisen@mpi-sb.mpg.de
[2] Universitá di Tor Vergata, Dipartimento di Ingegneria dell'Impresa, via del Politecnico 1, 00165, Roma, Italy
oriolo@disp.com
[3] Ecole Polytechnique Fédérale de Lausanne, SB-IMA-ROSO Station 8, CH-1015 Lausanne Switzerland
gautier.stauffer@epfl.ch
[4] Istituto di Analisi dei Sistemi ed Informatica del CNR, Viale Manzoni 30 - 00185 Roma, Italy
ventura@iasi.rm.cnr.it

Abstract. It is a long standing open problem to find an explicit description of the stable set polytope of *claw-free graphs*. Yet more than 20 years after the discovery of a polynomial algorithm for the maximum stable set problem for claw-free graphs, there is even no conjecture at hand today.

Such a conjecture exists for the class of *quasi-line graphs*. This class of graphs is a proper superclass of line graphs and a proper subclass of claw-free graphs for which it is known that not all facets have 0/1 normal vectors. *Ben Rebea's conjecture* states that the stable set polytope of a quasi-line graph is completely described by *clique-family* inequalities. Chudnovsky and Seymour recently provided a decomposition result for claw-free graphs and proved that Ben Rebea's conjecture holds, if the quasi-line graph is not a *fuzzy circular interval graph*.

In this paper, we give a proof of Ben Rebea's conjecture by showing that it also holds for fuzzy circular interval graphs. Our result builds upon an algorithm of Bartholdi, Orlin and Ratliff which is concerned with integer programs defined by circular ones matrices.

1 Introduction

A graph G is *claw-free* if no vertex has three pairwise nonadjacent vertices. Line graphs are claw free and thus the weighted stable set problem for a claw-free graph is a generalization of the weighted matching problem of a graph. While the general stable set problem is NP-complete, it can be solved in polynomial time on a claw-free graph [21, 29] even in the weighted case [22, 23] see also [32]. These algorithms are extensions of Edmonds' [10, 9] matching algorithms.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 291–305, 2005.

The stable set polytope $STAB(G)$ is the convex hull of the characteristic vectors of stable sets of the graph G. The polynomial *equivalence of separation and optimization* for rational polyhedra [16, 26, 18] provides a polynomial time algorithm for the separation problem for $STAB(G)$, if G is claw-free. However, this algorithm is based on the ellipsoid method [19] and no explicit description of a set of inequalities is known that determines $STAB(G)$ in this case. This apparent asymmetry between the algorithmic and the polyhedral status of the stable set problem in claw-free graphs gives rise to the challenging problem of providing a "...decent linear description of $STAB(G)$" [17], which is still open today. In spite of results characterizing the rank-facets [12] (facets with 0/1 normal vectors) of claw-free graphs, or giving a compact lifted formulation for the subclass of distance claw-free graphs [27] , the structure of the general facets for claw-free graphs is still not well understood and even no conjecture is at hand.

The matching problem [9] is a well known example of a combinatorial optimization problem in which the optimization problem on the one hand and the facets on the other hand are well understood. This polytope can be described by a system of inequalities in which the coefficients on the left-hand-side are 0/1. This property of the matching polytope does *not* extend to the polytope $STAB(G)$ associated with a claw-free graph. In fact, Giles and Trotter [14] show that for each positive integer a, there exists a claw-free graph G such that $STAB(G)$ has facets with $a/(a+1)$ normal vectors. Furthermore they show that there exist facets whose normal vectors have up to 3 different coefficients (indeed up to 5 as it is shown in [20]). Perhaps this is one of the reasons why providing a description of $STAB(G)$ is not easy, since 0/1 normal vectors can be interpreted as subsets of the set of nodes, whereas such an interpretation is not immediate if the normal vectors are not 0/1.

A graph is *quasi-line*, if the neighborhood of any vertex partitions into two cliques. The complement of quasi-line graphs are called *near-bipartite*, and a linear description of their stable set polytope has been given in [33]. The class of quasi-line graphs is a proper superclass of line graphs and a proper subclass of the class of claw-free graphs. Interestingly also for this class of graphs there are facets with $a/(a+1)$ normal vectors, for any nonnegative integer a [14], but no facet whose normal vector has more than 2 different coefficients is known for this class. Ben Rebea [28] considered the problem to study $STAB(G)$ for quasi-line graphs. Oriolo [25] formulated a conjecture inspired from his work.

Ben Rebea's Conjecture

Let $\mathcal{F} = \{K_1, \ldots, K_n\}$ be a set of cliques, $1 \leq p \leq n$ be integral and $r = n \mod p$. Let $V_{p-1}(\mathcal{F}) \subseteq V(G)$ the set of vertices covered by exactly $(p-1)$ cliques of $\mathcal{F}$ and $V_{\geq p}(\mathcal{F}) \subseteq V(G)$ the set of vertices covered by p or more cliques of $\mathcal{F}$. The inequality

$$(p-r-1) \sum_{v \in V_{p-1}(\mathcal{F})} x(v) + (p-r) \sum_{v \in V_{\geq p}(\mathcal{F})} x(v) \leq (p-r) \left\lfloor \frac{n}{p} \right\rfloor \qquad (1)$$

is valid for $STAB(G)$ and is called the *clique family inequality* associated with $\mathcal{F}$ and p.

Conjecture 1 (Ben Rebea's conjecture [25]). The stable set polytope of a quasi-line graph $G = (V, E)$ may be described by the following inequalities:

(i) $x(v) \geq 0$ for each $v \in V$
(ii) $\sum_{v \in K} x(v) \leq 1$ for each maximal clique K
(iii) inequalities (1) for each family $\mathcal{F}$ of maximal cliques and each integer p with $|\mathcal{F}| > 2p \geq 4$ and $|\mathcal{F}| \mod p \neq 0$.

In this paper we prove that Ben Rebea's Conjecture holds true. This is done by establishing the conjecture for *fuzzy circular interval graphs*, a class introduced by Chudnovsky and Seymour [6]. This settles the result, since Chudnovsky and Seymour showed that the conjecture holds if G is quasi-line and not a fuzzy circular interval graph. Interestingly, since all the facets are rank for this latter class of graphs, the quasi-line graphs that "produce" non-rank facets are the fuzzy circular interval graphs.

We first show that we can focus our attention on *circular interval graphs* [6] a subclass of fuzzy circular interval graphs. The weighted stable set problem over a circular interval graph may be formulated as a packing problem $\max\{c\,x \mid Ax \leq b, x \in \mathbb{Z}^n_{\geq 0}\}$, where $b = \mathbf{1}$ and $A \in \{0,1\}^{m \times n}$ is a *circular ones matrix*, i.e., the columns of A can be permuted in such a way that the ones in each row appear consecutively. Here the last and first entry of a row are also considered to be consecutive. Integer programs of this sort with general right-hand side $b \in \mathbb{Z}^m$ have been studied by Bartholdi, Orlin and Ratliff [3]. From this, we derive a separation algorithm which is based on the computation of a negative cycle, thereby extending a recent result of Gijswijt [13]. We then concentrate on packing problems with right-hand side $b = \alpha\,\mathbf{1}$, where α is an integer. By studying non-redundant cycles leading to separating hyperplanes, we show that each facet of the convex hull of integer feasible solutions to a packing problem of this sort has a normal vector with two consecutive coefficients. Instantiating this result with the case where $\alpha = 1$, we obtain our main result.

Cutting Planes

Before we proceed, we would like to stress some connections of this work to cutting plane theory. An inequality $c\,x \leq \lfloor \delta \rfloor$ is a *Gomory-Chvátal cutting plane* [15, 7] of a polyhedron $P \subseteq \mathbb{R}^n$, if $c \in \mathbb{Z}^n$ is an integral vector and $c\,x \leq \delta$ is valid for P. The *Chvátal closure* P^c of P is the intersection of P with all its Gomory-Chvátal cutting planes. If P is rational, then P^c is a rational polyhedron [30]. The separation problem for P^c is NP-hard [11]. A polytope P has *Chvátal-rank* one, if its Chvátal closure is the integer hull P_I of P. Let $QSTAB(G)$ be the *fractional stable set polytope* of a graph G, i.e., the polytope defined by non-negativity and clique inequalities. It is known [25] that $QSTAB(G)$ does not have Chvátal rank one, if G is a quasi-line graph. A famous example of a polytope of Chvátal-rank one is the fractional matching polytope and thus $QSTAB(G)$, where G is a line graph.

An inequality $c\,x \leq \delta$ is called a *split cut* [8] of P if there exists an integer vector $\pi \in \mathbb{Z}^n$ and an integer π_0 such that $c\,x \leq \delta$ is valid for $P \cap \{x \in \mathbb{R}^n \mid \pi\,x \leq \pi_0\}$ and for $P \cap \{x \in \mathbb{R}^n \mid \pi\,x \geq \pi_0 + 1\}$. The *split closure* P^s of P is the intersection of P with all its split cuts and this is a rational polyhedron if P itself is rational [8, 2]. The separation problem for the split closure is also NP-hard [4]. A polyhedron $P \subseteq \mathbb{R}^n$ has *split-rank* one, if $P^s = P_I$.

Both cutting plane calculi are simple procedures to derive valid inequalities for the integer hull of a polyhedron. It is easy to see that a clique family inequality is a split cut for $QSTAB(G)$ with $\pi(v) = 1$ if $v \in V_{p-1} \cup V_{\geq p}$, $\pi(v) = 0$ otherwise and $\pi_0 = \lfloor \frac{n}{p} \rfloor$. Thus, while the fractional stable set polytope of a quasi-line graph does not have Chvátal rank one, its split-rank is indeed one.

2 From Circular Interval to Quasi-Line Graphs

A circular interval graph [6] $G = (V, E)$ is defined by the following construction: Take a circle $\mathcal{C}$ and a set of vertices V on the circle. Take a subset of intervals $\mathcal{I}$ of $\mathcal{C}$ and say that $u, v \in V$ are adjacent if $\{u, v\}$ is a subset of one of the intervals.

Any interval used in the construction will correspond to a clique of G. Denote the family of cliques stemming from intervals by $\mathcal{K}_\mathcal{I}$ and the set of all cliques in G by $K(G)$. Without loss of generality, the (intervals) cliques of $\mathcal{K}_\mathcal{I}$ are such that none includes another. Moreover $\mathcal{K}_\mathcal{I} \subseteq K(G)$ and each edge of G is contained in a clique of $\mathcal{K}_\mathcal{I}$. Therefore, if we let $A \in \{0,1\}^{m \times n}$ be the clique vertex incidence matrix of $\mathcal{K}_\mathcal{I}$ and V one can formulate the the (weighted) stable set problem on a circular interval graph as a packing problem

$$\begin{aligned} \max \textstyle\sum_{v \in V} c(v)x(v) & \\ A\,x &\leq \mathbf{1} \\ x(v) &\in \{0,1\}\ \forall v \in V \end{aligned}$$

where the matrix A is a circular ones matrix (e.g. using clockwise ordering of the vertices).

Chudnovsky and Seymour [6] also introduced the more general class of *fuzzy circular interval graphs*. A graph G is a fuzzy circular interval if the following conditions hold.

(i) There is a map Φ from V to a circle $\mathcal{C}$.
(ii) There is a set of intervals $\mathcal{I}$ of $\mathcal{C}$, none including another, such that no point of C is the end of more than one interval so that:
  (a) If two vertices u and v are adjacent, then $\Phi(u)$ and $\Phi(v)$ belong to a common interval.
  (b) If two vertices u and v belong to a same interval, which is not an interval with endpoints $\Phi(u)$ and $\Phi(v)$, then they are adjacent.

In other words, in a fuzzy circular interval graph, adjacencies are completely described by the pair $(\Phi, \mathcal{I})$, except for vertices u and v such that one of the intervals with endpoints $\Phi(u)$ and $\Phi(v)$ belongs to $\mathcal{I}$. For these vertices adjacency

is fuzzy. We are particularly interested in *non-empty* cliques arising from endpoints of intervals of $\mathcal{I}$. If $[p, q]$ is an interval of $\mathcal{I}$ such that $\Phi^{-1}(p)$ and $\Phi^{-1}(q)$ are both non-empty, then we call the cliques $(\Phi^{-1}(p), \Phi^{-1}(q))$ a *fuzzy pair*.

Trivially, a circular interval graph is a fuzzy circular interval graph. When is a fuzzy circular interval graph a circular interval graph? The following lemma addresses this question. Say that a graph is C_4*-free* if it does not have an induced subgraph isomorphic to a cordless cycle of length 4. For $X \subseteq V$, we denote by $G[X]$ the subgraph of G *induced by* X.

Lemma 1 (Chudnovsky and Seymour [5]). *Let G be a fuzzy circular interval graph. If for every fuzzy pair of cliques (K_i, K_j), the subgraph $G[K_i \cup K_j]$ is C_4-free, then G is a circular interval graph.*

We sketch a proof of the above lemma. Trivially, if a fuzzy circular interval graph admits a *fuzzy representation* with no fuzzy pairs of cliques, then G is a circular interval graph. Now let $(\Phi, \mathcal{I})$ be a fuzzy representation of G minimizing the number of vertices belonging to fuzzy pairs. Let (K_1, K_2) be a fuzzy pair of cliques with respect to $(\Phi, \mathcal{I})$. Every vertex $v \in K_1$ has a neighbor and a non-neighbor in K_2. Otherwise one could remove v from the fuzzy pair, contradicting the minimality of $(\Phi, \mathcal{I})$. Now the following statement holds true: if $G = (V, E)$ is any graph with $V = V_1 \cup V_2$, V_1 and V_2 cliques and such that every vertex of V_1 (V_2) has a neighbor and a non-neighbor in V_2 (V_1), then there exist an induced $C_4 = \{u_1, u_2, v_2, v_1\}$ with $u_1, u_2 \in V_1$ and $v_1, v_2 \in V_2$.

Theorem 1. *Let F be a facet of $STAB(G)$, where G is a fuzzy circular interval graph. Then F is also a facet of $STAB(G')$, where G' is a circular interval graph obtained from G by removing some edges.*

Proof. Suppose that F is induced by the valid inequality $a\,x \leq \beta$. Trivially, if we remove an edge (u, v) connecting two vertices $u \in K_i$ to $v \in K_j$ of a fuzzy pair of cliques (K_i, K_j), the graph $G \setminus (u, v)$ is still a fuzzy circular interval graph. An edge e is F-critical, if $a\,x \leq \beta$ is not valid for $STAB(G \setminus e)$. If e is not F-critical, then F is also a facet of $STAB(G \setminus e)$. We prove that the removal of *all* non F-critical edges connecting two vertices in different cliques of fuzzy pairs results in a circular interval graph G'. Therefore, since F is a facet of $STAB(G')$, the claim follows.

Suppose G' is not a circular interval graph. Then from Lemma 1, there exists a fuzzy pair of cliques (K_1, K_2) such that the subgraph $G'[K_1 \cup K_2]$ contains a C_4. Say $V(C_4) = \{u_1, u_2, v_1, v_2\}$ with $u_1, u_2 \in K_1$, $v_1, v_2 \in K_2$, $(u_1, v_1), (u_2, v_2) \in E(C_4)$. The edge (u_1, v_1) is F-critical. Hence there exists a set S containing u_1, v_1 such that S violates $a\,x \leq \beta$ and S is stable in $G' \setminus (u_1, v_1)$. Property (ii) above implies that K_1 has no other fuzzy pair than K_2 and thus u_1 and u_2 are adjacent to the same vertices in $G' \setminus K_2$. This implies that $(S \setminus u_1) \cup u_2$ is a stable set. Therefore $a(u_2) < a(u_1)$ (else (u_1, v_1) is not F-critical). Applying the same argument to (u_2, v_2) leads to $a(u_1) < a(u_2)$. Which is a contradiction. □

Fuzzy circular interval graphs are quasi-line graphs. Chudnovsky and Seymour [6] gave a complete characterization of the stable set polytope of a quasi-line graph for the case in which the graph is *not* a fuzzy circular interval graph.

Let $\mathcal{F} = \{K_1, K_2, ..., K_{2n+1}\}$ an odd set of cliques of G. Let $T \subseteq V$ be the set of vertices which are covered by at least two cliques of $\mathcal{F}$. Then the inequality $\sum_{v \in T} x(v) \leq n$ is a valid inequality for $STAB(G)$ and inequalities of this type are called Edmonds' inequalities.

Theorem 2 ([6]). *Let G be a connected quasi-line graph, which is not a fuzzy circular interval graph. Then all non trivial facets of $STAB(G)$ are Edmonds' inequalities.*

Observe that Edmonds' inequalities are special clique family inequalities associated with $\mathcal{F}$ and $p = 2$. Moreover, Theorem 1 implies that, if we prove that all the non-trivial facets of the stable set polytope of a *circular interval graph* are clique family inequalities, then the same holds for the stable set polytope of a *fuzzy* one (cliques of G' are also cliques of G). Therefore, if we combine these facts, we may give a positive answer to the Ben Rebea's Conjecture if we prove that it holds for circular interval graphs. This is what we are going to show in the following sections.

3 Slicing and Separation

Let P be a polytope $P = \{x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0\}$, where $A \in \{0,1\}^{m \times n}$ is a circular ones matrix and $b \in \mathbb{Z}^m$ an integral vector. In this section, we consider the separation problem for the integer hull P_I of P:

> Given $x^* \in \mathbb{R}^n$, determine, whether $x^* \in P_I$ and if not, determine an inequality $c\,x \leq \delta$ which is valid for P_I and satisfies $c\,x^* > \delta$.

We present a membership algorithm of Gijswijt [13] and develop it further to retrieve a separating hyperplane. Following Bartholdi, Orlin and Ratliff [3], we consider the unimodular transformation $x = T\,y$, where T is the unimodular matrix

$$T = \begin{pmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & -1 & & & \\ & & & \ddots & & \\ & & & & 1 & \\ & & & & -1 & 1 \end{pmatrix} \tag{2}$$

The problem then reads, separate $y^* = T^{-1}x^*$ from the integer hull Q_I of the polytope Q defined by the system

$$\begin{pmatrix} A \\ -I \end{pmatrix} T\,y \leq \begin{pmatrix} b \\ 0 \end{pmatrix}. \tag{3}$$

In the following we denote the inequality system (3) by $B\,y \leq d$. Let us rewrite the matrix B as $B = (N|v)$, i.e. v is the n-th column of B. Observe that, by construction, v is also the last column of $\begin{pmatrix} A \\ -I \end{pmatrix}$.

Each row of the matrix N has at most one entry which is $+1$ and at most one entry which is -1. All other entries are 0. The matrix N is thus totally

unimodular. Thus, whenever $y(n)$ is set to an integer $\beta \in \mathbb{Z}$, the possible values for the variables $y(1), \ldots, y(n-1)$ define an integral polytope $Q_\beta = \{y \in \mathbb{R}^n \mid B\,y \leq d,\ y(n) = \beta\}$. We call this polytope Q_β the *slice* of Q defined by β.

Since T is unimodular, the corresponding slice of the original polyhedron $P \cap \{x \in \mathbb{R}^n \mid \sum_{i=1}^{n} x(i) = \beta\}$ is an integral polyhedron. From this it is already easy to see that the split-rank of P is one. However, we present a combinatorial separation procedure for the integer hull P_I of P which computes a split cut via the computation of a negative cycle.

If $y^*(n)$ is integral, then y^* lies in Q_I if and only if $y^* \in Q_{y^*(n)}$. Therefore we assume in the following that $y^*(n)$ is not integral and let β be an integer such that $\beta < y^*(n) < \beta + 1$ and let $1 > \mu > 0$ be the real number with $y^*(n) = \beta + 1 - \mu$. Furthermore, let Q_L and Q_R be the left slice Q_β and right slice $Q_{\beta+1}$ respectively. A proof of the next lemma follows from basic convexity.

Lemma 2. *The point y^* lies in Q_I if and only if there exist $y_L \in Q_L$ and $y_R \in Q_R$ such that*

$$y^* = \mu\, y_L + (1-\mu) y_R.$$

In the following we denote by $\overline{y} \in \mathbb{R}^{n-1}$ the vector of the first $n-1$ components of $y \in \mathbb{R}^n$. From the above discussion one has $y^* \in Q_I$ if and only if the following linear constraints have a feasible solution.

$$\begin{array}{r} \overline{y^*} = \overline{y_L} + \overline{y_R} \\ N\overline{y_L} \leq \mu\, d_L \\ N\overline{y_R} \leq (1-\mu)\, d_R \end{array}, \tag{4}$$

where $d_L = d - \beta v$ and $d_R = d - (\beta+1)v$.

Using Farkas' Lemma [31], it follows that equation (4) defines a feasible system, if and only if $\sum_{i=1}^{n-1} \lambda(i) y^*(i) + \mu f_L d_L + (1-\mu) f_R d_R$ is nonnegative, whenever λ, f_L and f_R satisfy

$$\begin{array}{r} \lambda + f_L N = 0 \\ \lambda + f_R N = 0 \\ f_L, f_R \geq 0. \end{array} \tag{5}$$

Now $\lambda + f_L N = 0$ and $\lambda + f_R N = 0$ is equivalent to $\lambda = -f_L N$ and $f_L N = f_R N$. Thus (4) defines a feasible system, if and only if the optimum value of the following linear program is nonnegative

$$\begin{array}{c} \min -f_L N\, \overline{y^*} + \mu f_L d_L + (1-\mu) f_R d_R \\ \\ \begin{array}{r} f_L N = f_R N \\ f_L, f_R \geq 0. \end{array} \end{array} \tag{6}$$

Let w be the negative sum of the columns of N. Then (6) is the problem of finding a minimum cost circulation in the directed graph $D = (U, \mathcal{A})$ defined by the edge-node incidence matrix

$$M = \begin{pmatrix} N & w \\ -N & -w \end{pmatrix} \text{ and edge weights } \mu(-N\,\overline{y^*} + d_L), (1-\mu)(-N\,\overline{y^*} + d_R) \tag{7}$$

Thus $y^* \notin Q_I$ if and only if there exists a negative cycle in $D = (U, \mathcal{A})$. The membership problem for Q_I thus reduces to the problem of detecting a negative cycle in D, see [13].

A separating split cut for y^* is an inequality which is valid for Q_L and Q_R but not valid for y^*. The inequality $f_L N\overline{y} \leq f_L d_L$ is valid for Q_L and the inequality $f_R N\overline{y} \leq f_R d_R$ is valid for Q_R. The corresponding disjunctive inequality (see, e.g., [24]) is the inequality

$$f_L N\overline{y} + c(n)y(n) \leq \delta, \text{ where } c(n) = f_L d_L - f_R d_R \text{ and } \delta = (\beta+1)f_L d_L - \beta f_R d_R. \tag{8}$$

The polytopes Q_L and Q_R are defined by the systems

$$\begin{array}{rl} y(n) = \beta & \\ N\,\overline{y} + v\,y(n) \leq d & \end{array} \qquad \text{and} \qquad \begin{array}{rl} y(n) = \beta + 1 & \\ N\,\overline{y} + v\,y(n) \leq d & \end{array} \tag{9}$$

respectively.

Let $f_{L,0}$ be the number $c(n) - f_L v$. Then the inequality (8) can be derived from the system defining Q_L with the weights $(f_{L,0}, f_L)$. Notice that, if y^* can be separated from Q_I, then $f_{L,0}$ must be positive. This is because y^* violates (8) and satisfies the constraints (9) on the left, where the equality $y(n) = \beta$ in the first line is replaced with $y(n) \geq \beta$. Let $f_{R,0}$ be the number $c(n) - f_R v$. Then the inequality (8) can be derived from the system defining Q_R with the weights $(f_{R,0}, f_R)$. Notice that, if y^* can be separated from Q_I, then $f_{R,0}$ must be negative.

A negative cycle in a graph with m edges and n nodes can be found in time $O(m\,n)$, see, e.g. [1]. Translated back to the original space and to the polyhedron P this gives the following theorem.

Theorem 3. *The separation problem for P_I can be solved in time $O(m\,n)$. Moreover, if $x^* \in P$ and $x^* \notin P_I$ one can compute in $O(m\,n)$ a split cut $c\,x \leq \delta$ which is valid for P_I and separates x^* from P_I together with a negative integer $f_{R,0}$, a positive integer $f_{L,0}$ and a vector f_L, f_R, which is the incidence vector of a simple negative cycle of the directed graph $D = (U, \mathcal{A})$ with edge-node incidence matrix and weights as in* (7)*, such that $c\,x \leq \delta$ is derived with from the systems*

$$\begin{array}{l} \mathbf{1}x \leq \beta \\ Ax \leq b \\ -x \leq 0. \end{array} \qquad \textit{and} \qquad \begin{array}{l} -\mathbf{1}x \leq -(\beta+1) \\ Ax \leq b \\ -x \leq 0, \end{array} \tag{10}$$

with the weights $f_{L,0}, f_L$ and $|f_{R,0}|, f_R$ respectively.

The above theorem gives an explicit derivation of the separating hyperplane as a split cut of P. We have the following corollary.

Corollary 1. *The integer hull P_I is the split closure of P.*

4 The Facets of P_I for the Case $b = \alpha \cdot 1$

In this section we study the facets of P_I, where $P = \{x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0\}$, where A is a circular ones matrix and b is an integer vector of the form $\alpha\,\mathbf{1}$, $\alpha \in \mathbb{N}$. For this, we actually inspect how the facets of the transformed polytope Q described in Section 3 are derived from the systems (9) and apply this derivation to the original system. It will turn out that the facet normal-vectors of P_I have only two integer coefficients, which are in addition consecutive. Since the stable set polytope of a circular interval graph is defined by such a system with $\alpha = 1$, we can later instantiate the results of this section to this special case. We can assume that the rows of A are inclusion-wise maximal.

Let F be a facet of Q_I and let y^* be in the relative interior of F. This facet F is generated by the unique inequality (8), which corresponds to a simple cycle of (6) of weight 0. Furthermore assume that F is not induced by an inequality $y(n) \leq \gamma$ for some $\gamma \in \mathbb{Z}$. Since F is a facet of the convex hull of integer points of two consecutive slices, we can assume that $y^*(n) = \beta + 1/2$ and thus that $\mu = 1/2$ in (6). This allows us to rewrite the objective function of problem (6) as follows:

$$\min(s^* + \tfrac{1}{2}v)\, f_L + (s^* - \tfrac{1}{2}v)\, f_R \tag{11}$$

where s^* is the slack vector

$$s^* = \begin{pmatrix} \alpha\,\mathbf{1} \\ \mathbf{0} \end{pmatrix} - By^* = \begin{pmatrix} \alpha\,\mathbf{1} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} A \\ -I \end{pmatrix} x^* \geq \mathbf{0}. \tag{12}$$

The point x^* in (12) is $x^* = T\,y^*$. Notice that x^* satisfies the system $A\,x \leq \alpha\,\mathbf{1}$.

Furthermore, we are interested in the facets of Q_I which are not represented by the system $B\,y \leq d$. If F is such a facet, then one can translate y^* away from Q_I, without changing $y^*(n) = \beta + 1/2$, such that $y^* \notin Q_I$ and $By^* \leq d$ with the property that the facet we are considering is the unique inequality (8), where f_L, f_R is a simple negative cycle in the graph $D = (U, \mathcal{A})$.

In the following we denote $U = \{1, \ldots, n\}$, where node i corresponds to the i-th column of the matrix M in (7). Notice that $\mathcal{A}$ partitions in two classes of arcs $\mathcal{A}_L$ and $\mathcal{A}_R$. The arcs $\mathcal{A}_R$ are simply the reverse of the arcs $\mathcal{A}_L$. $\mathcal{A}_L$ consists of two sets of arcs $\mathcal{S}_L$ and $\mathcal{T}_L$, where $\mathcal{S}_L$ is the set of arcs associated with inequalities $Ax \leq \alpha\,\mathbf{1}$ and $\mathcal{T}_L$ are the arcs stemming from the lower bounds $x \geq 0$. Likewise $\mathcal{A}_R$ can be partitioned into $\mathcal{S}_R$ and $\mathcal{T}_R$. In other words, if we look at the arc-node incidence matrix M in (7), the rows of M appear in the order $\mathcal{S}_L, \mathcal{T}_L, \mathcal{S}_R, \mathcal{T}_R$.

In particular, let a denote a row vector of A. Since A is a circular ones matrix one has $a\,x \leq \alpha \equiv \sum_{h=0}^{p} x(i+h) \leq \alpha$ for some suitable i and p, where computation is modulo n, so $x_n \equiv x_0$, $x_{n+1} \equiv x_1$, etc. It is straightforward to see that $a\,x \leq \alpha$ generates the arcs $(i+p, i-1) \in \mathcal{S}_L$ and $(i-1, i+p) \in \mathcal{S}_R$ of A, see Figure 1. The weights of the two arcs coincide, if $n \notin \{i, i+1, \ldots, i+p\}$ and is exactly the slack $\alpha - \sum_{h=0}^{p} x^*(i+h)$ in this case. Otherwise, the weight of the arc $(i+p, i-1)$ is $\alpha - \sum_{h=0}^{p} x^*(i+h) + 1/2$ and the weight of the arc $(i-1, i+p)$ is $\alpha - \sum_{h=0}^{p} x^*(i+h) - 1/2$.

On the other hand, a lower bound $-x_i \leq 0$ generates the two arcs $(i-1,i) \in \mathcal{T}_L$ and $(i,i-1) \in \mathcal{T}_R$. The weight of both arcs is equal to $x^*(i)$, if $i \neq n$. If $i = n$, the arc $(n-1,n) \in \mathcal{T}_L$ has weight $x^*(n) - 1/2$ and $(n,n-1) \in \mathcal{T}_R$ has weight $x^*(n) + 1/2$.

Since the slacks are non-negative, the arcs whose cost is equal to the corresponding slack minus $\frac{1}{2}$ are the only candidates to have a negative cost. We call those *light* arcs. Consequently we call those arcs whose cost is equal to the slack plus $\frac{1}{2}$ *heavy*. Observe that the light arcs belong to $\mathcal{S}_R \cup \{(n-1,n)\}$.

Lemma 3. *Let $\mathcal{C}$ be a simple negative cycle in D, then the following holds:*

(a) $\mathcal{C}$ contains strictly more light arcs than heavy ones.
(b) An arc of $\mathcal{C}$ in $\mathcal{S}_L$ ($\mathcal{T}_L$) cannot be immediately followed or preceded by an arc in $\mathcal{S}_R$ ($\mathcal{T}_R$).
(c) The cycle $\mathcal{C}$ contains at least one arc of $\mathcal{S}_R$ or contains no arc of $\mathcal{S}_L \cup \mathcal{S}_R$.

Proof. (a) follows from the fact that the slacks are nonnegative. (b) follows from our assumption that the rows of the matrix A are maximal and that $\mathcal{C}$ is simple.

To prove (c) suppose that the contrary holds. It follows that $(n-1,n)$ is in $\mathcal{C}$, because it is the only light arc not in $\mathcal{S}_R$. We must reach $n-1$ on the cycle without using heavy arcs.

Each arc in $\mathcal{S}_L$ with starting node n is heavy. Thus $(n-1,n)$ is followed by $(n,1) \in \mathcal{T}_L$. Suppose that $(n-1,n)$ is followed by a sequence of arcs in $\mathcal{T}_L$ leading to i and let $(i,j) \notin \mathcal{T}_L$ be the arc which follows this sequence. It follows from (b) that $(i,j) \notin \mathcal{T}_R$ and thus that $(i,j) \in \mathcal{S}_L$. Since (i,j) cannot be heavy, we have $1 \leq j < i < n$. This is a contradiction to the fact that $\mathcal{C}$ is simple, since we have a sub-cycle contained in $\mathcal{C}$, defined by (i,j) and $(j,j+1),\ldots,(i-1,i)$. □

Lemma 4. *If there exists a simple cycle $\mathcal{C}$ of D with negative cost, then there exists a simple cycle $\mathcal{C}'$ of D with negative cost that does not contain any arc from $\mathcal{S}_L$.*

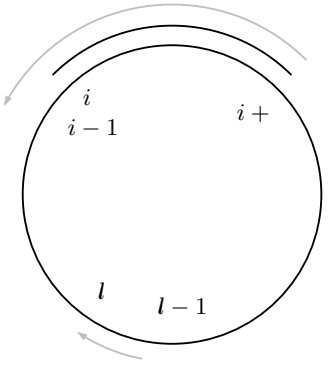


Fig. 1. The incidence vector of a row of A consists of the nodes$\{i, i+1, \ldots, i+\ \}$ which are consecutive on the cycle in clockwise order. Its corresponding arc in $\mathcal{S}_L$ is the arc $(i+\ \ , i-1)$. The arc $(l-1,l)$ in $\mathcal{T}_L$ corresponds to the lower bound $\ (l) \geq 0$

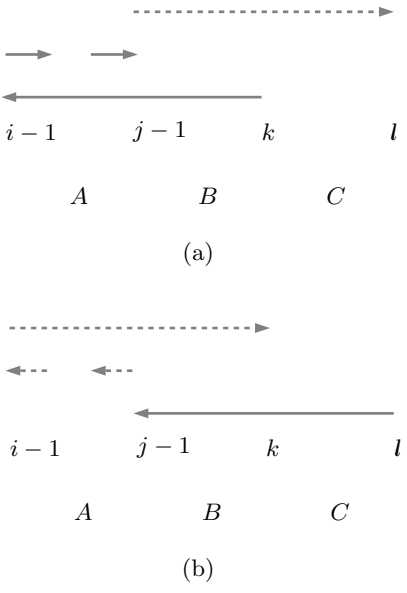


Fig. 2. (a) depicts an arc $(k, i-1) \in \mathcal{S}_L$, followed by arcs in $\mathcal{T}_L$ and the arc $(j-1, l) \in \mathcal{S}_R$. (b) depicts the situation, where the intermediate arcs are in $\mathcal{T}_R$

Proof. Suppose that $\mathcal{C}$ also contains an arc from the set $\mathcal{S}_L$. We know from Lemma 3 that the cycle $\mathcal{C}$ contains at least one arc of $\mathcal{S}_R$. Lemma 3 implies that $\mathcal{C}$ has an arc in $\mathcal{S}_L$, followed by arcs in $\mathcal{T}_L$ or $\mathcal{T}_R$ but not both, followed by an arc in $\mathcal{S}_R$. We first consider the case that the intermediate arcs are all in $\mathcal{T}_L$.

This situation is depicted in Figure 2, (a). The arc in $\mathcal{S}_L$ is $(k, i-1)$. This is followed by the arcs $(i-1, i), \ldots, (i-1, j-1)$ in $\mathcal{T}_L$ and the arc $(j-1, l)$ in $\mathcal{S}_R$. Let this be the path $\mathcal{P}_1$. We now show that we can replace this path with the path $\mathcal{P}_2 = (k, k+1), \ldots, (l-1, l)$ consisting of arcs in $\mathcal{T}_L$. We proceed as follows. First we show that the weight of this path is at most the weight of the original path, where we ignore the addition of $\pm 1/2$ to the arc-weights. Let light($\mathcal{P}$) and heavy($\mathcal{P}$) be the number of light and heavy edges in a path $\mathcal{P}$, respectively. We then show that light($\mathcal{P}_2$) − heavy($\mathcal{P}_2$) = light($\mathcal{P}_1$) − heavy($\mathcal{P}_1$), from which we can conclude the claim in this case.

Consider the set of indices $\mathcal{A} = \{i, \ldots, j-1\}$, $\mathcal{B} = \{j, \ldots, k\}$ and $\mathcal{C} = \{k+1, \ldots, l\}$ and the numbers $A = \sum_{\mu \in \mathcal{A}} x^*(\mu)$, $B = \sum_{\mu \in \mathcal{B}} x^*(\mu)$ and $C = \sum_{\mu \in \mathcal{C}} x^*(\mu)$. Ignoring the eventual addition of $\pm 1/2$ to the edge weights, we have that the weight of $\mathcal{P}_2$ is C and that of $\mathcal{P}_1$ is $\alpha - (A+B) + A + \alpha - (B+C)$ and suppose that this is less than C. Then $B + C > \alpha$ which is not possible, since x^* satisfies the constraints $Ax \leq \alpha \mathbf{1}$. Thus, if none of the edges in $\mathcal{P}_1$ and $\mathcal{P}_2$ is heavy or light, the weight of $\mathcal{P}_2$ is at most the weight of $\mathcal{P}_1$.

Suppose now that $n \in \mathcal{A}$. Then $\mathcal{P}_1$ contains exactly one heavy edge $(k, i-1)$ and one light edge $(n-1, n)$. The path $\mathcal{P}_2$ contains no heavy or light edge. Suppose that $n \in \mathcal{B}$, then $\mathcal{P}_1$ contains exactly one heavy edge, $(k, i-1)$ and one

light edge $(j-1,l)$. $\mathcal{P}_2$ does not contain a heavy or light edge. If $n \in \mathcal{C}$, then $\mathcal{P}_1$ contains exactly one light edge $(j-1,l)$ and no heavy edge. $\mathcal{P}_2$ also contains exactly one light edge $(n-1,n)$. This concludes the claim for the case that an arc of $\mathcal{S}_L$ is followed by arcs of $\mathcal{T}_L$ and an arc of $\mathcal{S}_R$.

The case, where the intermediate arcs belong to $\mathcal{T}_R$ is depicted in Figure 2, (b). The assertion follows by a similar argument. □

Combining Theorem 3 with the above lemma we obtain the following theorem.

Theorem 4. *Let $P = \{x \in \mathbb{R}^n \mid Ax \leq \alpha\mathbf{1}, x \geq 0\}$ be a polyhedron, where $A \in \{0,1\}^{m\times n}$ is a circular ones matrix and $\alpha \in \mathbb{N}$ a positive integer. A facet of P_I is of the form*

$$a \sum_{v\in T} x(v) + (a-1) \sum_{v\notin T} x(v) \leq a\,\beta, \tag{13}$$

where $T \subseteq \{1,\ldots,n\}$ and $a, \beta \in \mathbb{N}$.

Proof. Theorem 3 implies that a facet which is not induced by $Ax \leq \alpha\,\mathbf{1}$, $x \geq 0$ or $\mathbf{1}\,x \leq \gamma$ is a nonnegative integer combination of the system on the left in (10) with nonnegative weights $f_{L,0}, f_L$. Lemma 4 implies that f_L can be chosen such that the only nonzero $(+1)$ entries of f_L are corresponding to lower bounds $-x(v) \leq 0$. The theorem thus follows with $a = f_{0,L}$ and T set to those variables, whose lower bound inequality does not appear in the derivation. □

5 The Solution to Ben Rebea's Conjecture

Let G be a circular interval graph and let $\mathcal{K}_\mathcal{I}$ the family of cliques stemming from the intervals in the definition of G (see Section 2). Then $QSTAB(G) = \{x \in R^n \mid Ax \leq \mathbf{1}, x \geq 0\}$ where the 0/1 matrix A, corresponding to the cliques $\mathcal{K}_\mathcal{I}$, has the circular ones property. Theorem 4 implies that any facet of $STAB(G)$ is of the form

$$a \sum_{v\in T} x(v) + (a-1) \sum_{v\notin T} x(v) \leq a \cdot \beta \tag{14}$$

We now show that a facet, which is not induced by an inequality of $Ax \leq \mathbf{1}, x \geq 0$ is induced by a clique family inequality associated with some set of cliques $\mathcal{F} \subseteq \mathcal{K}_I$ and some integer p. Recall from Theorem 3 that any facet of this kind can be derived from the system

$$\begin{aligned} -\mathbf{1}x &\leq -(\beta+1) \\ Ax &\leq \mathbf{1} \\ -x &\leq 0, \end{aligned} \tag{15}$$

with weights $|f_{R,0}|, f_R$, where $f_{R,0}$ is a negative integer while f_R is a 0-1 vector. A *root* of F is a stable set, whose characteristic vector belongs to F. In particular, we have that the multiplier $f_R(v)$ associated with a lower bound $-x(v) \leq 0$ must

be 0 if v belongs to a root of size $\beta+1$. If v does not belong to a root of size β or to a root of size $\beta+1$, then the facet is induced by $x(v) \geq 0$. Thus if $v \notin T$, then v belongs to a root of size $\beta+1$.

Let $\mathcal{F} = \{K \in \mathcal{K}_I \mid f_R(K) \neq 0\}$ and $p = a + |f_{R,0}|$. The multiplier $|f_{R,0}|$ must satisfy

$$\begin{array}{rl} -|f_{R,0}| + |\{K \in \mathcal{F} \mid v \in K\}| = a-1 & \forall v \notin T \\ -|f_{R,0}| + |\{K \in \mathcal{F} \mid v \in K\}| = a & \forall v \in T,\ v \text{ is in a root of size } \beta+1 \\ -|f_{R,0}| + |\{K \in \mathcal{F} \mid v \in K\}| \geq a & \forall v \in T,\ v \text{ is not in a root of size } \beta+1 \\ -|f_{R,0}|(\beta+1) + |\mathcal{F}| = a\beta & \end{array}$$

Observe that $|\mathcal{F}| = (a + |f_{R,0}|)\beta + |f_{R,0}|$ and therefore $r = |\mathcal{F}| \mod p = |f_{R,0}|$. Moreover, any vertex not in T belongs to exactly $p-1$ cliques from $\mathcal{F}$, while each vertex in T belongs to at least p cliques from $\mathcal{F}$. Therefore, inequality (14) is the clique family inequality associated with $\mathcal{F}$ and p. We may therefore state the following theorem.

Theorem 5. *Let G be a circular interval graph. Then any facet of $STAB(G)$, which is not induced by an inequality of the system $Ax \leq \mathbf{1}$, $x \geq 0$, is a clique family inequality associated with some $\mathcal{F}$ and p such that $|\mathcal{F}| \mod p \neq 0$.*

If we combine this result with Theorem 1, Theorem 2 and we recall that Edmonds' inequalities are also clique family inequalities associated with $|\mathcal{F}|$ odd and $p = 2$, we obtain the following corollary.

Corollary 2. *Let G be a quasi-line graph. Any non-trivial facet of $STAB(G)$ is a clique family inequality associated with some $\mathcal{F}$ and p such that $|\mathcal{F}| \mod p \neq 0$.*

We may assume, without loss of generality, that the cliques in the family $\mathcal{F}$ are maximal [25]. A last lemma is the missing brick to the solution of Ben Rebea's conjecture.

Lemma 5. *Let G be a quasi-line graph and $(\mathcal{F}, p)$ a pair such that*

$$(p-r-1) \sum_{v \in V_{p-1}(\mathcal{F})} x(v) + (p-r) \sum_{v \in V_{\geq p}(\mathcal{F})} x(v) \leq (p-r) \left\lfloor \frac{|\mathcal{F}|}{p} \right\rfloor \qquad (16)$$

is a facet of $STAB(G)$. If $|\mathcal{F}| < 2p$, then the inequality (16) is a clique inequality.

We may therefore state our main result:

Theorem 6. *Ben Rebea's conjecture holds true.*

Acknowledgements

We are in debt to Maria Chudnovsky for communicating Lemma 1 to us. We also thank Maria Chudnovsky, Tom Liebling, Antonio Sassano, Paul Seymour, Bruce Shepherd, and Leslie Trotter for several useful discussions, support and encouragement. Many thanks are also due to the anonymous referees for their very helpful comments and remarks.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. Prentice Hall Inc., Englewood Cliffs, NJ, 1993. Theory, algorithms, and applications.
2. K. Andersen, G. Cornuéjols, and Y. Li. Split closure and intersection cuts. In *Integer programming and combinatorial optimization*, volume 2337 of *Lecture Notes in Comput. Sci.*, pages 127–144. Springer, Berlin, 2002.
3. J. J. Bartholdi, J. B. Orlin, and H. Ratliff. Cyclic scheduling via integer programs with circular ones. *Operations Research*, 28:1074–1085, 1980.
4. A. Caprara and A. N. Letchford. On the separation of split cuts and related inequalities. *Mathematical Programming. A Publication of the Mathematical Programming Society*, 94(2-3, Ser. B):279–294, 2003. The Aussois 2000 Workshop in Combinatorial Optimization.
5. M. Chudnovsky. Personal communication, 2004.
6. M. Chudnovsky and P. Seymour. The structure of claw-free graphs. In *Proceedings of the Bristish Combinatorial Conference, Durham*, 2005. to appear.
7. V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.
8. W. Cook, R. Kannan, and A. Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47:155–174, 1990.
9. J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards*, 69:125–130, 1965.
10. J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
11. F. Eisenbrand. On the membership problem for the elementary closure of a polyhedron. *Combinatorica*, 19(2):297–300, 1999.
12. A. Galluccio and A. Sassano. The rank facets of the stable set polytope for claw-free graphs. *Journal on Combinatorial Theory*, 69:1–38, 1997.
13. D. Gijswijt. On a packet scheduling problem for smart antennas and polyhedra defined by circular ones matrices. submitted to *Siam Journal of Discrete Mathematics*, 2003.
14. R. Giles and L. E. T. jun. On stable set polyhedra for $k_{(1,3)}$-free graphs. *Journal on Combinatorial Theory*, 31:313–326, 1981.
15. R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.
16. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.
17. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer Verlag, Berlin, 1988.
18. R. M. Karp and C. H. Papadimitriou. On linear characterizations of combinatorial optimization problems. *SIAM Journal on Computing*, 11(4):620–632, 1982.
19. L. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1097, 1979.
20. T. M. Liebling, G. Oriolo, B. Spille, and G. Stauffer. On the non-rank facets of the stable set polytope of claw-free graphs and circulant graphs. *Mathematical Methods of Operations Research*, 59:25–35, 2004.
21. L. Lovász and M. Plummer. *Matching Theory*. North Holland, Amsterdam, 1986.
22. G. J. Minty. On maximal independent sets of vertices in claw-free graphs. *Journal on Combinatorial Theory*, 28:284–304, 1980.

23. D. Nakamura and A. Tamura. 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):194–2004, 2001.
24. G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley, 1988.
25. G. Oriolo. Clique family inequalities for the stable set polytope for quasi-line graphs. *Discrete Applied Mathematics*, 132(3):185–201, 2003.
26. M. W. Padberg and M. R. Rao. The Russian method for linear programming III: Bounded integer programming. Technical Report 81-39, New York University, Graduate School of Business and Administration, 1981.
27. W. R. Pulleyblank and F. B. Shepherd. Formulations for the stable set polytope of a claw-free graph. In G. Rinaldi and L. Wolsey, editors, *Proceedings Third IPCO Conference*, pages 267–279, 1993.
28. A. B. Rebea. *Étude des stables dans les graphes quasi-adjoints*. PhD thesis, Université de Grenoble, 1981.
29. N. Sbihi. Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. *Discrete Mathematics*, 29:53–76, 1980.
30. A. Schrijver. On cutting planes. *Annals of Discrete Mathematics*, 9:291–296, 1980.
31. A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley, 1986.
32. A. Schrijver. *Combinatorial optimization. Polyhedra and efficiency (3 volumes)*. Algorithms and Combinatorics 24. Berlin: Springer., 2003.
33. F. B. Shepherd. Applying Lehman's theorems to packing problems. *Mathematical Programming*, 71:353–367, 1995.

Approximation Algorithms for Stochastic Inventory Control Models

Retsef Levi[1,⋆], Martin Pál[2,⋆⋆], Robin Roundy[3,⋆⋆⋆], and David B. Shmoys[4,†]

[1] School of ORIE, Cornell University,
Ithaca, NY 14853
rl227@cornell.edu
[2] DIMACS Center, Rutgers University,
Piscataway, NJ 08854-8018
mpal@dimacs.rutgers.edu
[3] School of ORIE, Cornell University,
Ithaca, NY 14853
robin@orie.cornell.edu
[4] School of ORIE and Department of Computer Science,
Cornell University,
Ithaca, NY 14853
shmoys@cs.cornell.edu

Abstract. We consider stochastic control inventory models in which the goal is to coordinate a sequence of orders of a single commodity, aiming to supply stochastic demands over a discrete finite horizon with minimum expected overall ordering, holding and backlogging costs. In particular, we consider the *periodic-review stochastic inventory control problem* and the *stochastic lot-sizing problem* in the case where demands over time are correlated and non-stationary (time-dependent). In these cases, it is usually very hard to compute the optimal policy. We provide what we believe to be the first computationally efficient policies with constant worst-case performance guarantees. More specifically, we provide a general 2-approximation algorithm for the periodic-review stochastic inventory control problem and a 3-approximation algorithm for the stochastic lot-sizing problem.

Our approach is based on several novel ideas: we present a new (marginal) cost accounting for stochastic inventory models; we use cost-balancing techniques; and we consider non base-stock (order-up-to) policies that are extremely easy to implement on-line. Our results are valid for all of the currently known approaches in the literature to model correlation and non-stationarity of demands over time.

⋆ Research supported partially by a grant from Motorola and NSF grants CCR-9912422& CCR-0430682.

⋆⋆ Research supported by ONR grant N00014-98-1-0589.

⋆⋆⋆ Research supported partially by a grant from Motorola, NSF grant DMI-0075627, and the Querétaro Campus of the Instituto Tecnológico y de Estudios Superiores de Monterrey.

† Research supported partially by NSF grants CCR-9912422& CCR-0430682.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 306–320, 2005.

1 Introduction

In this paper we address the long-standing problem of finding computationally efficient and provably good inventory control policies in supply chains with correlated and non-stationary (time-dependent) stochastic demands. This problem arises in many domains and has many practical applications such as dynamic forecast updates (for some applications see [5, 8]). We consider two classical models, the *periodic-review stochastic inventory control problem* and the *stochastic lot-sizing problem* with correlated and non-stationary demands. Here the correlation is inter-temporal, i.e., what we observe in period s changes our forecast for the demand in future periods. We provide what we believe to be the first computationally efficient policies with constant worst-case performance guarantees; that is, there exists a constant C such that, for any given joint distribution of the demands, the expected cost of the policy is guaranteed to be within C times the expected cost of an optimal policy. More specifically, we provide a worst-case performance guarantee of 2 for the periodic-review stochastic inventory control problem, and a performance guarantee of 3 for the stochastic lot-sizing problem.

The Models. The details of the periodic-review stochastic inventory control problem are as follows. A sequence of random demands for a single commodity at a single location occurs over a finite planning horizon of T discrete periods. The random demands over the T periods are denoted by $D_1, \ldots, D_T$, and can be non-stationary and correlated. The goal is to coordinate a sequence of orders over the planning horizon aiming to satisfy these demands with minimum expected cost. In each period we can order any number of units that are assumed to arrive only after L periods, where L is a positive integer that is called the *lead time*. The cost is incurred in the following way. At the beginning of each period t (for $t = 1, \ldots, T$) we first receive the order that was placed at time period $t - L$. Next we place an order (for any number of units). Let q_t be the size of the order placed in period t. This order will incur a per-unit ordering cost c_t for each unit ordered, i.e., an overall cost of $c_t q_t$ (this order will arrive in period $t + L$). We then observe the realized demand d_t of the random demand D_t (as a convention throughout the paper we will use capital letters to denote random variables and lower case letter to denote a realization of them). Each demand can be supplied only from physical on-hand inventory. Let NI_t be the *net inventory* at the beginning of period t. The net inventory can be positive in case we have excess (physical) on-hand inventory, or negative in case there is a shortage, i.e., there are units of demands still waiting to be satisfied. Unsatisfied units of demand are called *back orders*. More specifically, we can fully supply the demand d_t only if we have enough physical inventory, i.e., $ni_t + q_{t-L} \geq d_t$. The assumption is that unsatisfied units of demand are going to stay in the system until they are satisfied. Hence, at the end of each period t we incur one of the following costs. If the net inventory at the end of the period, $NI_{t+1} = ni_t + q_{t-L} - D_t$, is positive, i.e., $NI_{t+1} > 0$, we incur a per-unit *holding cost*, h_t, for each excess unit that is carried in inventory from period t to period $t + 1$ (an overall cost of $h_t NI_{t+1}$).

On the other hand, if there is a shortage, i.e., $NI_{t+1} < 0$, we incur a per-unit backlogging penalty cost, p_t, for each unit that is still back ordered (the overall cost is $p_t(-NI_{t+1})$). The only assumption on the cost parameters $c_t,\ h_t,\ p_t$ is that they are non-negative, and that there is *no speculative motivation* to hold inventory. In particular, in the case where there is no lead time ($L = 0$), this implies that if we know that a demand is going to occur in period t, it is always cheaper to order it in period t rather than $t-1$ or $t+1$ (i.e., $c_{t-1} + h_{t-1} \geq c_t$ and $c_{t+1} + p_t \geq c_t$). This assumption on the cost parameters implies that, without loss of generality, one can assume that $c_t = 0$ and $h_t,\ p_t \geq 0$ for each $t = 1, \ldots, T$. More specifically, any instance of the problem that satisfies the property of no speculative motivation specified above, can be converted to an equivalent instance with the property that $c_t = 0$ and $h_t,\ p_t \geq 0$ for each t (i.e., the modified instance has the same set of optimal policies). This is done by means of a cost transformation commonly used in the inventory literature. The details of this cost transformation will be discussed at the end of Section 3. In the stochastic lot-sizing problem, we consider, in addition, a fixed ordering cost that is incurred in each period in which an order is placed (regardless of its size), but we assume that there is no lead time ($L = 0$).

In both models, the goal is to find a policy of orders with minimum expected overall discounted cost over the given planning horizon, where the cost incurred in period t is discounted by α^t ($0 < \alpha \leq 1$). Since the costs are non-stationary we assume, again without loss of generality, that $\alpha = 1$.

The assumptions that we make on the demand distributions are very mild and generalize all of the currently known approaches in the literature to model correlation and non-stationarity of demands over time (for details we refer the reader to [7, 11, 17]). As part of the model, we will assume that at the beginning of each period s, we are given what we call an *information set* that is denoted by f_s. The information set f_s contains all of the information that is available at the beginning of time period s. More specifically, the information set f_s consists of the realized demands $(d_1, \ldots, d_{s-1})$ over the interval $[1, s)$, and possibly some more (external) information denoted by $(w_1, \ldots, w_s)$ (e.g., w_j can specify the next period in which new information on the demand distribution is going to become available to us). The information set f_s in period s is one specific realization in the set of all possible realizations of the random vector $(D_1, \ldots, D_{s-1}, W_1, \ldots, W_s)$. This set is denoted by $\mathcal{F}_s$. In addition, we assume that in each period s there is a known conditional joint distribution of the future demands $(D_s, \ldots, D_T)$, denoted by $I_s := I_s(f_s)$, which is determined by f_s (i.e., if we know f_s, we also know $I_s(f_s)$). For ease of notation, D_t will always denote the random demand in period t according to the conditional joint distribution I_s for some $s \leq t$, where it will be clear from the context to which period s we refer. We will use t as the general index for time, and s will always refer to the period we are currently in. The only assumption on the demands is that for each $s = 1, \ldots, T$, and each $f_s \in \mathcal{F}_s$ the conditional expectation $E[D_t | f_s]$ is well defined and finite for each period $t \geq s$.

We note again that by allowing correlation we let I_s be dependent on the realization of the demands over the periods $1, \ldots, s-1$ and possibly on some other information that becomes available by time s (i.e., I_s is a function of f_s). However, the conditional joint distribution I_s is assumed to be independent of the specific inventory control policy being considered. Moreover, in period s, each feasible policy can only use the information f_s, i.e., we allow only *non-anticipatory* policies.

Related Literature. These models have attracted the attention of many researchers over the years and there exists a huge body of related literature. The dominant paradigm in almost all of the existing literature has been to formulate these models using a dynamic programming framework. The optimization problem is defined recursively over time using subproblems for each possible state of the system. If all the demands are independent, then the state consists of a given time period s and the *inventory position* at the beginning of the period, which is the net inventory at the beginning of the period plus all the units that were ordered over the last L periods, i.e., $ni_s + \sum_{j=s-L}^{s-1} q_j$. If the demands are correlated then the state will also consist of a given information set $f_s \in \mathcal{F}_s$. For each subproblem, we compute an optimal solution to minimize the expected overall discounted cost from time s until the end of the horizon, assuming that we make optimal decisions in future periods.

This framework has turned out to be very effective in characterizing the structure of optimal policies for the overall system. Surprisingly, the optimal policies for these rather complex models follow simple forms, known as *state-dependent base-stock policies*. For each period s and observed information set $f_s \in \mathcal{F}$, there exists an optimal target base-stock level that is independent of the starting inventory position level at the beginning of the period (i.e., it is only a function of s and f_s). The optimal policy aims to keep the inventory level at each period as close as possible to the target base-stock level. That is, it orders up to the target level whenever the inventory level at the beginning of the period is below that level, and orders nothing otherwise (see [11, 7, 25, 17] for a discussion of different results along these lines).

Unfortunately, these rather simple forms of policies do not always lead to efficient algorithms for computing the optimal policies. This is especially true in the presence of correlated and non-stationary demands which cause the state space of the relevant dynamic programs to grow exponentially and explode very fast. The difficulty essentially comes from the fact that we need to solve 'too many' subproblems. This phenomenon is known as *the curse of dimensionality*. Moreover, because of this phenomenon, it seems unlikely that there exists an efficient algorithm to solve these huge dynamic programs.

Muharremoglu and Tsitsiklis [16] have proposed an alternative approach to the dynamic programming framework. They have observed that this problem can be decoupled into a series of *unit supply-demand subproblems*, where each subproblem corresponds to a single unit of supply and a single unit of demand that are matched together. This novel approach enabled them to substantially simplify some of the dynamic programming based proofs on the structure of op-

timal policies, as well as to prove several important new structural results. Using this unit decomposition, they have also suggested new methods to compute the optimal policies. However, their computational methods are essentially dynamic programming approaches applied to the unit subproblems, and hence they suffer from similar problems in the presence of correlated and non-stationary demand. Although our approach is very different than theirs, we use some of their ideas as technical tools in some of the proofs in the paper.

As a result of this apparent computational intractability, many researchers have attempted to construct computationally efficient (but suboptimal) heuristics for these problems. However, we are aware of very few attempts to analyze the worst-case performance of these heuristics (see for example [12]). Moreover, we are aware of no computationally efficient policies for which there exist constant performance guarantees. For details on some of the proposed heuristics and a discussion of others, see [11, 12, 7]. One specific class of suboptimal policies that has attracted a lot of attention is the class of *myopic policies*. In a myopic policy, in each period we attempt to minimize the expected cost for that period, ignoring the impact on the cost in future periods. The myopic policy is attractive since it yields a base-stock policy that is easy to compute on-line, that is, it does not require information on the control policy in the future periods. In many cases, the myopic policy seems to perform well. However, in many other cases, especially when the demand can drop significantly from one period to the next, the myopic policy performs poorly and can even be arbitrarily more expensive than the optimal policy (see [9]). Myopic policies were extensively explored by Vienott [24, 23], Ignall and Veinott [6], Zipkin [25], Iida and Zipkin [7] and Lu, Song and Regan [12].

Chan and Muckstadt [2] have considered a different way for approximating huge dynamic programs that arise in the context of inventory control problems. More specifically, they have considered un capacitated and capacitated multi-item models. Instead of solving the one period problem (as in the myopic policy) they have added to the one period problem a penalty function which they call Q-function. This function accounts for the holding cost incurred by the inventory left at the end of the period over the entire horizon. Their look ahead approach with respect to the holding cost is somewhat related to our approach, though significantly different.

Another approach that was applied to these models is the robust optimization approach (see [1]). Here the assumption is of a distribution-free model, where instead the demands are assumed to be drawn from some specified uncertainty set. Each policy is then evaluated with respect to the worst possible sequence of demands within the given uncertainty set. The goal is to find the policy with the best worst-case (i.e., a min-max approach). This objective is very different from the objective of minimizing expected (average cost) discussed in most of the existing literature, including this paper.

We note that our work is also related to a huge body of approximation results for stochastic and on-line combinatorial problems. The work on approximation results for stochastic combinatorial problems goes back to the work of Möhring,

Radermacher and Weiss [13, 14] and the more recent work of Möhring, Schulz and Uetz [15]. They have considered stochastic scheduling problems. However, their performance guarantees are dependent on the specific distributions (namely on second moment information). Recently, Dean, Goemans and Vondrák considered a version of a multi-stage stochastic knapsack problem [3]. There is also a growing stream of approximation results for several 2-stage stochastic combinatorial problems. For a comprehensive literature review we refer the reader to [22, 4, 20]. We note that the problems we consider in this paper are by nature *multi-stage* stochastic problems, which are usually much harder.

Our Work. Our work is distinct from the existing literature in several significant ways, and is based on three novel ideas:

Marginal cost accounting. We introduce a novel approach for cost accounting in stochastic inventory control problems. The key observation is that once we place an order of a certain number of units in some period, then the expected ordering and holding cost that these units will incur over the rest of the planning horizon is a function only of the realized demands over the rest of the horizon, not of future orders. Hence, with each period, we can associate the overall expected ordering and holding cost that is incurred by the units ordered in this period, over the entire horizon. We note that similar ideas of holding cost accounting were previously used in the context of models with continuous time, infinite horizon and stationary (Poisson distributed) demand (see, for example, the work of Axsäter and Lundell [19] and Axsäter in [18]). However, this new way of marginal cost accounting is significantly different from the dynamic programming approach, which, in each period, accounts only for the costs that are incurred in that period. We believe that this new approach will have more applications in the future in analyzing stochastic inventory control problems.

Cost balancing. The idea of cost balancing was used in the past to construct heuristics with constant performance guarantees for deterministic inventory problems. The most well-known example is the *part-period cost balancing heuristic* of Silver and Meal for the lot-sizing problem (see [21]). We are not aware of any application of these ideas to stochastic inventory control problems.

Non base-stock policies. Our policies are not state-dependent base-stock policies. This enable us to use, in each period, the distributional information about the future demands beyond the current period (unlike the myopic policy), without the burden of solving huge dynamic programs. Moreover, our policies can be easily implemented on-line and are simple, both conceptually and computationally.

Using these ideas we provide what is called a 2-approximation algorithm for the periodic-review stochastic inventory control problem; that is, the expected cost of our policy is no more than twice the expected cost of an optimal policy. Our result is valid for all known approaches used to model correlated and non-stationary demands. We also note that this guarantee refers only to the worst-case performance and it is likely that on average the performance would be significantly better. We then use the standard cost transformation (mentioned

above) to achieve significantly better guarantees if the ordering cost is the dominant part in the overall cost, as it is the case in many real life situations. For the stochastic lot-sizing problem we provide a 3-approximation algorithm. This is again a worst-case analysis and we would expect the typical performance to be much better.

The rest of the paper is organized as follows. In Section 2 we explain the details the marginal cost accounting approach. In Section 3 we describe a 2-approximation algorithm for the periodic-review stochastic inventory control problem and possible extensions. The stochastic lot-sizing problem is briefly discussed in Section 4. A full version of this extended abstract can be found in [9]. In subsequent work [10], we were able to construct a 2-approximation algorithm for the periodic-review stochastic inventory control problem with capacity constraints on the size of the order in each period.

2 Marginal Cost Accounting

In this section, we will present a new approach to the cost accounting of stochastic inventory control problems. Our approach differs from the traditional dynamic programming based approach. In particular, we account for the holding cost incurred by a feasible policy in a different way, which enables us to design and analyze new approximation algorithms.

Recall that the inventory position is equal to the net inventory plus all the units in orders that are on the way. Let X_s be the inventory position at the beginning of period s before the order in that period was placed, and let Y_s be the inventory position after the order in period s was placed. Similarly, x_s and y_s will denote the realization of X_s and Y_s, respectively. Observe that once we decide to order q_s units in period s (where $q_s = y_s - x_s$), then the holding cost they are going to incur from period s until the end of the planning horizon is independent of any future decision in subsequent time periods. It is dependent only on the demand to be realized over the time interval $[s, T]$.

To make this rigorous, we use a *ground distance-numbering scheme* for the units of demand and supply, respectively. More specifically, we think of two infinite lines, each starting at 0, the *demand line* and the *supply line*. The demand line $\mathcal{L}_D$ represents the units of demands that can be potentially realized over the planning horizon, and similarly, the supply line $\mathcal{L}_S$ represents the units of supply that can be ordered over the planning horizon. Each 'unit' of demand, or supply, now has a *distance-number* according to its respective distance from the origin of the demand line and the supply line, respectively. If we allow continuous demand (rather then discrete) and continuous order quantities the unit and its distance-number are defined infinitesimally. We can assume without loss of generality that the units of demands are realized according to increasing distance-number. Similarly, we can describe each policy P in terms of the periods in which it orders each supply unit, where all unordered units are "ordered" in period $T+1$. It is also clear that we can assume without loss of generality that the supply

units are ordered in increasing distance-number. We can further assume (again without loss of generality) that as the demand is realized, the units of supply are consumed on a *first-ordered-first-consumed basis*. Therefore, we can *match* each unit of supply that is ordered to a certain unit of demand that has the same number. We note that Muharremoglu and Tsitsiklis [16] have used the idea of matching units of supply to units of demand in a novel way to characterize and compute the optimal policy in different stochastic inventory models. However, their computational method is based on applying dynamic programming to the single-unit problems. Therefore, their cost accounting within each single-unit problem is still additive, and differs fundamentally from ours.

Suppose now that at the beginning of period s we have observed information set f_s. Observe that for each non-anticipatory policy, knowing the observed information set in period s implies that the inventory position at the beginning of period s is known deterministically. Let x_s be the inventory position at the beginning of period s, and suppose that q_s additional units are ordered. We have already seen that, without loss of generality, we can assume that the supply units in inventory are going to be consumed on a *first-ordered-first-consumed basis*. This implies that the expected additional (marginal) holding cost that these q_s units are going to incur from time period s until the end of the planning horizon is in independent of any future decision. More specifically, it is equal to $\sum_{j=s+L}^{T} E[h_j(q_s - (D_{[s,j]} - x_s)^+)^+|f_s]$ (recall that we assume without loss of generality that $\alpha = 1$), where $x^+ = \max(x, 0)$ and $D_{[s,j]}$ is the accumulated demand over the interval $[s, j]$, i.e., $D_{[s,j]} = \sum_{t=s}^{j} D_t$. We note again that the assumption that units of supply are consumed on a first-ordered-first-consumed basis, was used previously to compute holding costs in models with continuous time, infinite horizon and stationary demand (mainly Poisson distributed demand) [19, 18].

Using this approach, consider any feasible policy P and let $H_t^P := H_t^P(Q_t^P)$ $(t = 1, \ldots, T)$ be the discounted ordering and holding cost incurred by the additional Q_t^P units ordered in period t by policy P. Thus, $H_t^P = H_t^P(Q_t^P) := c_t Q_t^P + \sum_{j=t+L}^{T} h_j(Q_t^P - (D_{[t,j]} - X_t)^+)^+$. Now let B_t^P be the discounted backlogging cost incurred in period $t + L$ $(t = 1 - L, \ldots, T - L)$. In particular, $B_t^P := p_{t+L}(D_{[t,t+L]} - (X_{t+L} + Q_t^P))^+$ (where $D_j := 0$ with probability 1 for each $j \leq 0$, and $Q_t^P = q_t$ is given as part of the input for each $t \leq 0$). Let $\mathcal{C}(P)$ be the cost of the policy P. Clearly,

$$\mathcal{C}(P) := \sum_{t=1-L}^{0} B_t^P + H_{[1,L]} + \sum_{t=1}^{T-L} (H_t^P + B_t^P),$$

where $H_{[1,L]}$ denotes the total holding cost incurred over the interval $[1, L]$ (by units ordered before period 1). We note that the first two expressions $\sum_{t=1-L}^{0} B_t^P$ and $H_{[1,L]}$ are not affected by our decisions (i.e., they are the same for any feasible policy and each realization of the demand), and therefore we will omit them. Since they are non-negative, this will not effect our results. Also observe that without loss of generality, we can assume that $Q_t^P = H_t^P = 0$ for any

policy P and each period $t = T - L + 1, \ldots, T$, since nothing that is ordered in these periods can be used within the given planning horizon. We now can write $\mathcal{C}(P) = \sum_{t=1}^{T-L}(H_t^P + B_t^P)$. In some sense, we change the accounting of the holding cost from periodical to marginal, i.e., we associate with each decision of how many units to order in each period all the costs that, after this decision is made, become independent of any future decision. As we will demonstrate in the sections to come, this new approach serves as a powerful tool for designing simple policies with expected cost guaranteed to be within a constant factor of the optimal expected cost.

3 Dual-Balancing Policy

In this section, we consider a new policy for the periodic-review stochastic inventory control problem, which we call a *dual-balancing policy*. Recall the assumption on the cost parameters, in particular, that there is no speculative motivation for holding inventory or backorders. As mentioned in Section 1, this implies that, without loss of generality, we can assume that $c_t = 0$ and $h_t,\ p_t \geq 0$ for each $t = 1, \ldots, T$. Next we shall describe the new policy and its worst-case analysis under the above assumption. The generality of this assumption will be discussed in detail at the end of this section. In this policy we aim to balance the expected marginal holding cost against the expected marginal backlogging cost. In each period $s = 1, \ldots, T - L$, we focus on the units that we order in period s only, and balance the expected holding cost they are going to incur over $[s, T]$ against the expected backlogging cost in period $s + L$. We do that using the marginal accounting of the holding cost that was introduced in Section 2.

We next describe the details of the policy, which is very simple to implement, and then analyze its expected performance. A superscript B will refer to the dual-balancing policy described below.

The Policy. We first describe the policy and its worst-case analysis in the case where the demands have a density function and fractional orders are allowed. Later on, we will describe how to extend the policy and the analysis to the case in which the demands and the order sizes are integer-valued. In each period $s = 1, \ldots, T - L$, we consider a given information set f_s (where again $f_s \in \mathcal{F}_s$) and the resulting pair (x_s^B, I_s) of the inventory position at the beginning of period s and the conditional joint distribution I_s of the demands $(D_s, \ldots, D_T)$. We then consider the following two functions:

(i) The expected holding cost over $[s, T]$ that is incurred by the *additional* q_s units ordered in period s, conditioning on f_s. We denote this function by $l_s^B(q_s)$, where
$l_s^B(q_s) := E[H_s^B(q_s)|f_s]$ and $H_s^B(Q_s) := \sum_{j=s+L}^{T} h_j(Q_s - (D_{[s,j]} - X_s)^+)^+$.

(ii) The expected backlogging cost incurred in period $s + L$ as a function of the additional q_s units ordered in period s, conditioning again on f_s. We denote this function by $b_s^B(q_s)$, where $b_s^B(q_s) := E[B_s^B(q_s)|f_s]$ and
$B_s^B := p_s(D_{[s,s+L]} - (X_s^B + Q_s))^+ = p_s(D_{[s,s+L]} - Y_s^B)^+$. We note again

that conditioned on some $f_s \in \mathcal{F}_s$ and given any policy P, we already know x_s, the starting inventory position in time period s. Hence, the backlogging cost in period s, $B_s^B|f_s$, is indeed only a function of q_s and future demands.

The dual-balancing policy now orders q_s^B units in period s, where q_s^B is such that the expected holding cost incurred over the time interval $[s, T]$ by the additional q_s^B units we order at s is equal to the expected backlogging cost in period $s + L$. In other words, we set $l_s^B(q_s^B) = b_s^B(q_s^B)$ (i.e., $E[H_s^B(q_s^B)|f_s] = E[B_s^B(q_s^B)|f_s]$).

Since we assume that the demands are continuous we know that the functions $l_t^P(q_t)$ and $b_t^P(q_t)$ are continuous in q_t for each $t = 1, \ldots, T - L$ and each feasible policy P. It is straightforward to verify that both $l_s^P(q_s)$ and $b_s^P(q_s)$ are convex functions of q_s. Moreover, the function $l_s^P(q_s)$ is equal to 0 for $q_s = 0$ and is an increasing function in q_s, which goes to infinity as q_s goes to infinity. In addition, the function $b_s^P(q_s)$ is non-negative for $q_s = 0$ and is a decreasing function in q_s, which goes to 0 as q_s goes to infinity. Thus, q_s^B is well-defined and we can indeed balance the two functions. We also point out that q_s^B can be computed as the minimizer of the function $g_s(q_s^B) := \max\{l_s^B(q_s), b_s^B(q_s)\}$. Since $g_s(q_s)$ is the maximum of two convex functions of q_s, it is also a convex function of q_s. This implies that in each period s we need to solve a single-variable convex minimization problem and this can be solved efficiently. In particular, if for each $j \geq s$, $D_{[s,j]}$ has any of the distributions that are commonly used in inventory theory, then it is extremely easy to evaluate the functions $l_s^P(q_s)$ and $b_s^P(q_s)$. More generally, the complexity of the algorithm is of order T (number of time periods) times the complexity of solving the single variable convex minimization defined above. The complexity of this minimization problem can vary depending on the level of information we assume on the demand distributions and their characteristics. In all of the common scenarios there exist straightforward methods to solve this problem efficiently.

We end this discussion by pointing out that the dual-balancing policy is not a state-dependent base-stock policy. However, it can be implemented on-line, free from the burden of solving large dynamic programming problems. This concludes the description of the policy for continuous-demand case. Next we describe the analysis of the worst-case expected performance of this policy.

Analysis. We start the analysis by expressing the expected cost of the dual-balancing policy.

Lemma 1. *Let $\mathcal{C}(B)$ denote the cost of the dual-balancing policy. Then $E[\mathcal{C}(B)] = 2\sum_{t=1}^{T-L} E[Z_t]$, where $Z_t := E[H_t^B|\mathcal{F}_t] = E[B_t^B|\mathcal{F}_t]$ $(t = 1, \ldots, T - L)$.*

Proof. In Section 2, we have already observed that the cost $\mathcal{C}(B)$ of the dual-balancing policy can be expressed as $\sum_{t=1}^{T-L}(H_t^B + B_t^B)$. Using the linearity of expectations and conditional expectations, we can express $E[\mathcal{C}(B)]$ as

$$\sum_{t=1}^{T-L} E[E[(H_t^B(Q_t^B) + B_t^B(Q_t^B))|\mathcal{F}_t]].$$

However, by the construction of the policy, we know that for each $t = 1, \ldots, T - L$, we have that $E[H_t^B|\mathcal{F}_t] = E[B_t^B|\mathcal{F}_t] = Z_t$. Note that Z_t is a random variable and a function of the realized information set f_t in period t. We then conclude that the expected cost of the solution provided by the dual-balancing policy is $E[\mathcal{C}(B)] = 2\sum_{t=1}^{T-L} E[Z_t]$, where for each t, the expectation $E[Z_t]$ is taken over the possible realizations of information sets in period t, i.e., over the set $\mathcal{F}_t$.

Next we wish to show that the expected cost of any feasible policy is at least $\sum_{t=1}^{T-L} E[Z_t]$. In each period $t = 1, \ldots, T - L$ let $\mathcal{Q}_t \subseteq \mathcal{L}_S$ be the set of supply units that were ordered by the dual-balancing policy in period t.

Given an optimal policy OPT and the dual-balancing policy B, we define the following random variables Z'_t for each $t = 1, \ldots, T-L$. In case $Y_t^{OPT} \leq Y_t^B$, we let Z'_t be equal to the backlogging cost incurred by OPT in period $t+L$, denoted by B_t^{OPT}. In case $Y_t^{OPT} > Y_t^B$, we let Z'_t be the holding cost that the supply units in $\mathcal{Q}_t$ incur in OPT, denoted by $\bar{H}_t^{OPT}$. Note that by our assumption each of the supply units in $\mathcal{Q}_t$ was ordered by OPT in some period t' such that $t' \leq t$. Moreover, for each period s, if we condition on the some information set $f_s \in \mathcal{F}_s$ and given the two policies OPT and B, then we already know y_s^{OPT} and y_s^B deterministically, and hence we know which one of the above cases applies to Z'_s, but we still do not know its value.

The next lemma shows that $\sum_{t=1}^{T-L} E[Z'_t]$ is at most the expected cost of OPT, denoted by *opt*. In other words, it provides a lower bound on the expected cost of an optimal policy. Observe that this lower bound is closely related to the dual-balancing policy through the definition of the variables Z'_t (the proof is omitted due lack of space).

Lemma 2. *Given an optimal policy OPT, we have* $\sum_{t=1}^{T-L} E[Z'_t] \leq E[\mathcal{C}(OPT)] =: opt.$

Next we would like to show that for each $s = 1, \ldots, T - L$, we have that $E[Z_s] \leq E[Z'_s]$.

Lemma 3. *For each* $s = 1, \ldots, T - L$, *we have* $E[Z_s] \leq E[Z'_s]$.

Proof. First observe again that if we condition on some information set $f_s \in \mathcal{F}_s$, then we already know whether $Z'_s = B_s^{OPT}$ or $Z'_s = \bar{H}_s^{OPT}$. It is enough to show that for each possible information set $f_s \in \mathcal{F}_s$, conditioning on f_s, we have $z_s \leq E[Z'_s|f_s]$.

In case $Z'_s := B_s^{OPT}$, we know that $y_s^{OPT} \leq y_s^B$. Since for each period t and any given policy P, $B_t^P := p_t(D_{[t,t+L]} - Y_t)^+$, this implies that for each possible realization $d_s, \ldots, d_T$ of the demands over $[s, T]$, we have that the backlogging cost B_s^B that the dual-balancing policy will incur in period $s + L$ is at most the backlogging cost $B_s^{OPT} = Z'_s$ that OPT will incur in that period. That is, with probability 1, we have $B_s^B|f_s \leq Z'_s|f_s$. The claim for this case then follows immediately.

We now consider the case where $Z'_s := \bar{H}_s^{OPT}$. Observe again that each unit in $\mathcal{Q}_s$ was ordered by OPT in some period $t' \leq s$. Recall that for each period t,

the per unit ordering cost is equal to $c_t = 0$ and the per unit holding cost h_t is assumed to be nonnegative. It is then clear that for each realization of demands $d_s, \ldots, d_T$ over the interval $[s, T]$, the ordering and holding costs $H_s^B|f_s$ that the units in $\mathcal{Q}_s$ will incur in B will be at most the holding costs $\bar{H}_s^{OPT}|f_s = Z_s'|f_s$ that they will incur in OPT.

As a corollary of Lemmas 1, 2 and 3, we conclude the following theorem.

Theorem 1. *The dual-balancing policy provides a 2-approximation algorithm for the periodic-review stochastic inventory control problem with continuous demands and orders, i.e., for each instance of the problem, the expected cost of the dual-balancing policy is at most twice the expected cost of an optimal solution.*

3.1 Extensions

Integer-Valued Demands. We now briefly discuss the case in which the demands are integer-valued random variables, and the order in each period is also assumed to be integer. In this case, in each period s, the functions $l_s^B(q_s)$ and $b_s^B(q_s)$ are originally defined only for integer values of q_s. We define these functions for any value of q_s by interpolating piecewise linear extensions of the integer values. It is clear that these extended functions preserve the properties of convexity and monotonicity discussed in the previous (continuous) case. However, it is still possible (and even likely) that the value q_s^B that balances the functions l_s^B and b_s^B is not an integer. Instead we consider the two consecutive integers q_s^1 and $q_s^2 := q_s^1 + 1$ such that $q_s^1 \leq q_s^B \leq q_s^2$. In particular, $q_s^B := \lambda q_s^1 + (1 - \lambda) q_s^2$ for some $0 \leq \lambda \leq 1$. We now order q_s^1 units with probability λ and q_s^2 units with probability $1 - \lambda$. This constructs what we call a *randomized dual-balancing policy.* By rather straightforward arguments, one can establish an analogous theorem to Theorem 1 above, and show that the randomized dual-balancing policy provides a 2-approximation for this case.

Stochastic Lead Times. We can also consider a more general model, where the lead time of an order placed in period s is some integer-valued random variable L_s. However, we assume that the random variables $L_1, \ldots, L_T$ are correlated, and in particular, that, with probability 1, $s + L_s \leq t + L_t$ for each $s \leq t$. In other words, we assume that any order placed at time s will arrive no later than any other order placed after period s. This is a very common assumption in the inventory literature, usually described as "no order crossing". The algorithm and the analysis can then be extended to provide a 2-approximation algorithm for this more general case (see [9] for details).

3.2 Cost Transformation

In this section, we discuss in detail the cost transformation that enables us to assume, without loss of generality that, for each period $t = 1, \ldots, T$, we have $c_t = 0$ and $h_t,\ p_t \geq 0$. Consider any instance of the problem with cost parameters that imply no speculative motivation for holding inventory or backorders (as discussed

in Section 1). We use a transformation of the cost parameters, commonly used in the inventory theory literature. This constructs an equivalent instance with the property that, for each period $t = 1, \ldots, T$, we have $c_t = 0$ and $h_t,\ p_t \geq 0$. More specifically, the modified instance has the same optimal policies, since the expected cost of each feasible policy is changed by the same constant. Applying the dual-balancing policy to that instance will provide a feasible policy to the original instance that has a performance guarantee of at most 2.

Next we describe the transformation for the case with no lead time ($L = 0$) and $\alpha = 1$; the extension to the case of arbitrary lead time and α is straightforward. Recall that any feasible policy P satisfies the following equations. For each $t = 1, \ldots, T$, we have that $Q_t = NI_t - NI_{t-1} + D_t$. Using these equations we can express the ordering cost in each period t as $c_t(NI_t - NI_{t-1} + D_t)$. Using the positive and negative parts of the net inventory at the end of each period, we can replace NI_t with $NI_t^+ - NI_t^-$. This leads to the following transformation of cost parameters. We let $\hat{c}_t := 0,\ \hat{h}_t := h_t + c_t - c_{t+1}$ ($c_{T+1} = 0$) and $\hat{p}_t := p_t - c_t + c_{t+1}$. Note that the assumption that there is no speculative motivation to hold inventory or back orders, implies that $\hat{h}_t$ and $\hat{b}_t$ are non-negative ($t = 1, \ldots, T$). It is readily verified that the induced instance is equivalent to the original one, and that the expected cost of each feasible policy is *decreased* by $E[\sum_{t=1}^{T} c_t d_t]$. Hence, any 2-approximation algorithm for the modified instance will yield a 2-approximation algorithm for the original instance.

Moreover, in practice, it is often the case that the ordering cost is the dominant part of the overall cost. In particular, if $E[\sum_{t=1}^{T} c_t d_t]$ is large compared to the optimal expected cost, then applying the dual-balancing policy to the modified instance will yield a significantly better performance guarantee (see [9] for more details).

4 The Stochastic Lot-Sizing Problem

In this section, we change the previous model and in addition to the per-unit ordering cost, consider a fixed ordering cost K that is incurred in each period t with positive order (i.e., when $Q_t > 0$). We call this model the *stochastic lot-sizing problem*. The goal is again to find a policy that minimizes the expected discounted overall ordering, holding and backlogging costs. Here we will assume that $L = 0$ and that in each period $t = 1, \ldots, T$, the conditional joint distribution I_t of $(D_t, \ldots, D_T)$ is such that the demand D_t is known deterministically (i.e., with probability 1).

Next we briefly describe a policy which we call the *triple-balancing policy* and denote by TB. It can be shown that its worst-case expected performance guarantee is 3 (see [9]).

The policy follows two rules that specify when to place an order and how many units to order once an order is placed. In each period s, we consider the last period in which we have placed an order, say s^*. We then place an order only if by not placing it the backlogging cost over $(s^*, s]$ will exceed K. This is possible since at the beginning of period s the demand d_s is known deterministically. Once

an order is placed, we order q_s units where q_s is the maximum number of units, such that the expected holding cost that these units will incur until the end of the planning horizon (i.e., over $[s, T]$) is at most K.

The analysis in this case is based on the following idea. We consider all the random orders placed by the TB policy. These orders induced a random partition of the planning horizon into intervals. The analysis shows that over each interval, the expected cost of the TB policy is at most $3K$, whereas the expected cost of the optimal policy is at least K. We can then show that the performance guarantee of the policy is 3.

Theorem 2. *The Triple-Balancing policy provides a 3-approximation for the stochastic lot-sizing problem with correlated demands.*

Acknowledgments

We would to thank Shane Henderson for stimulating discussions and helpful suggestions that made the paper significantly better and to Jack Muckstadt for his valuable comments that lead to the extension to the stochastic lead times.

References

1. D. Bertsimas and A. Thiele. A robust optimization approach to supply chain management. In *Proceedings of 14th IPCO*, pages 86–100, 2004.
2. E. W. M. Chan. *Markov Chain Models for multi-echelon supply chains.* PhD thesis, School of OR&IE, Cornell University, Ithaca, NY, January 1999.
3. B. C. Dean, M. X. Goemans, and J. Vondák. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *Proceedings of the 45th Annual IEEE Symposium on the Foundations of Computer Science, 2004.*, pages 208–217, 2004.
4. S. Dye, L. Stougie, and A. Tomasgard. The stochastic single resource service provision problem. *Naval Research Logistics*, 50:869–887, 2003.
5. N. Erkip, W. H. Hausman, and S. Nahmias. Optimal centralized ordering policies in multi-echelon inventory systems with correlated demands. *Mangement Science*, 36:381–392, 1990.
6. E. Ignall and A. F. Venott. Optimality of myopic inventory policies for several substitute products. *Managment Science*, 15:284–304, 1969.
7. T. Iida and P. Zipkin. Approximate solutions of a dynamic forecast-inventory model. Working paper, 2001.
8. H. L. Lee, K. C. So, and C. S. Tang. The value of information sharing in two-level supply chains. *Mangement Science*, 46:626–643, 1999.
9. R. Levi, M. Pal, R. O. Roundy, and D. B. Shmoys. Approximation algorithms for stochastic inventory control models. Technical Report TR1412, ORIE Department, Cornell University, 2004. Submitted.
10. R. Levi, R. O. Roundy, D. B. Shmoys, and V. A. Troung. Approximation algorithms for capacitated stochastic inventory control models. Working paper, 2004.
11. D. Lingxiu and H. L. Lee. Optimal policies and approximations for a serial multi-echelon inventory system with time-correlated demand. *Operations Research*, 51, 2003.

12. X. Lu, J. S. Song, and A. C. Regan. Inventory planning with forecast updates: approximate solutions and cost error bounds. Working paper, 2003.
13. R. H. Möhring, F. J. Radermacher, and G. Weiss. Stochastic scheduling problems I: general strategies. *ZOR- Zeitschrift fur Operations Research*, 28:193–260, 1984.
14. R. H. Möhring, F. J. Radermacher, and G. Weiss. Stochastic scheduling problems II: set strategies. *ZOR- Zeitschrift fur Operations Research*, 29:65–104, 1984.
15. R. H. Möhring, A. Schulz, and M. Uetz. Approximation in stochastic scheduling: the power of LP-based priority policies. *Journal of the ACM (JACM)*, 46:924–942, 1999.
16. A. Muharremoglu and J. N. Tsitsiklis. A single-unit decomposition approach to multi-echelon inventory systems. Working paper, 2001.
17. Ö. Özer and G. Gallego. Integrating replenishment decisions with advance demand information. *Managment Science*, 47:1344–1360, 2001.
18. S. Axsäter. Simple solution procedures for a class of two-echelon inventory problems. *Operations Research*, 38:64–69, 1990.
19. S. Axsäter and P. Lundell. In process safety stock. In *Proceedings of the 23rd IEEE Conference on Decision and Control*, pages 839–842, 1984.
20. D. B. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *Proceedings of the 45th Annual IEEE Symposium on the Foundations of Computer Science, 2004.*, pages 228–237, 2004.
21. E. A. Silver and H. C. Meal. A heuristic selecting lot-size requirements for the case of a deterministic time varying demand rate and discrete opportunities for replenishment. *Production and Inventory Management*, 14:64–74, 1973.
22. L. Stougie and M. H. van der Vlerk. Approximation in stochastic integer programming. Technical Report SOM Research Report 03A14, Eindhoven University of Technology, 2003.
23. A. F. Veinott. Optimal policy for a multi-product, dynamic, non-stationary inventory problem. *Operations Research*, 12:206–222, 1965.
24. A. H. Veinott. Optimal policies with non-stationary demands. In H. E. Scarf, D. M. Gilford, and M. W. Shelly, editors, *Multistage inventory models and techniques*, pages 85–115. Stanford University Press, 1963.
25. P. H. Zipkin. *Foundation of inventory management.* The McGraw-Hill Companies, Inc, 2000.

On Two-Stage Stochastic Minimum Spanning Trees*

Kedar Dhamdhere[1], R. Ravi[2], and Mohit Singh[2]

[1] Dept. of Computer Science, Carnegie Mellon University,
Pittsburgh PA 15213
kedar@cs.cmu.edu
[2] Tepper School of Business, Carnegie Mellon University,
Pittsburgh PA 15213
{ravi, mohits}@andrew.cmu.edu

Abstract. We consider the undirected minimum spanning tree problem in a stochastic optimization setting. For the two-stage stochastic optimization formulation with finite scenarios, a simple iterative randomized rounding method on a natural LP formulation of the problem yields a nearly best-possible approximation algorithm.

We then consider the Stochastic minimum spanning tree problem in a more general black-box model and show that even under the assumptions of bounded inflation the problem remains log -hard to approximate unless $P = NP$; where is the size of graph. We also give approximation algorithm matching the lower bound up to a constant factor.

Finally, we consider a slightly different cost model where the second stage costs are independent random variables uniformly distributed between $[0, 1]$. We show that a simple thresholding heuristic has cost bounded by the optimal cost plus $\frac{\zeta(3)}{4} +$ (1).

1 Introduction

Stochastic optimization refers to problems where the inputs have an uncertainty, usually modeled by giving a probability distribution over the inputs. A common framework for stochastic optimization problem is two-stage stochastic optimization with recourse [2]. The uncertainty in the variables is modeled by probability distribution over a set of *scenarios* one of which will emerge tomorrow. *Recourse* is the ability to take corrective action tomorrow when one of the scenarios emerges.

This framework is well suited to network design problems where, in practice, the network has to be designed depending on the future demand patterns. One of the basic problems in network design is the minimum cost spanning tree problem. In this paper, we consider various models of the Stochastic Minimum Spanning Tree problem and give near optimal approximation algorithms for them.

* Supported in part by NSF grant CCR-0105548 and ITR grant CCR-0122581 (The ALADDIN project).

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 321–334, 2005.

Related Work. Stochastic Programming is well studied field with vast literature [16]. Recently, there has been work in designing approximation algorithm for the problems [6, 7, 9, 12, 15].

The models considered here are usually the two stage stochastic optimization with recourse. However, the variables in each scenario are correlated; e.g., in the network design problems considered by [6, 7, 9, 12], the costs of all edges in each scenario increases by the same factor.

In [15], Shmoys et al consider the problem of Stochastic set-cover and show how a ρ-approximation for the deterministic version of the problem can be used to obtain a 2ρ-approximation to the Stochastic version of the set-cover problem. Although, the spanning tree problem can be formulated as a set cover problem (see Section 3), the techniques of [15] do not yield a solution as the set cover problem is exponential in size. Also, the deterministic spanning tree problem when formulated as a set cover problem has a integrality gap of 2 while stochastic version of the problem is $O(\log n)$-hard to approximate unless $P = NP$. Hence, it is unlikely that techniques discussed in [15] will be applicable to Stochastic MST problem.

2 Models and Our Results

2.1 Stochastic MST with Explicit Scenarios

A popular formulation for stochastic optimization problems explicitly lists the finite set of scenarios which can occur tomorrow. The stochastic version of the minimum spanning tree problem under this model can be stated as follows: In the first stage we are given a complete graph $G = (V, E)$ on n nodes and a non-negative cost function c^0 on edges. In the second stage we have k scenarios and probabilities p^i, $1 \leq i \leq k$ of their occurrences. Each scenario has a different cost function c^i on the edges. The problem is select a subset of edges E^0 in the first stage and then augment it with E^i in the i^{th} scenario to obtain a spanning tree in each of the k scenarios. The objective function to minimize is the expected cost of the edges chosen. Note that, this cost is $c^0(E^0) + \sum_{i=1}^{k} p^i c^i(E^i)$. Observe that in this model the costs of the edges in any scenario need not be correlated. Although, the deterministic version of the problem has linear-time randomized algorithm [10], the stochastic version of this problem is hard. It has been shown in [3, 5] that the problem is hard to approximate within a factor ofmin$\{\log n$, $\log k\}$, by a reduction from set-cover.

We give the following result for this model:

Theorem 1. *There exists a polynomial time randomized algorithm which returns a solution of expected cost* $O((\log n + \log k) \cdot \mathrm{OPT})$ *with high probability, where* OPT *is the cost of the optimum solution of the Stochastic spanning tree problem on graph* $G = (V, E)$, $|V| = n$ *with* k *scenarios. The running time is polynomial in* k *and* n.

Observe that, when $k = poly(n)$ our algorithm gives the best approximation up to a constant factor.

We formulate a simple linear program for the stochastic spanning tree problem. We then use an optimal fractional solution to randomly round each edge. The techniques used for proving Theorem 1 are adapted from Alon[1].

2.2 Stochastic MST in the Black-Box Model

In this model, the scenarios are not stated explicitly but we have access to a Black-box from which we can sample the second stage scenarios. The samples are drawn from the same probability distribution as the second stage scenarios. Let $\lambda = max_{1 \leq i \leq k, e \in E}\{\frac{c_e^i}{c_e^0}, \frac{c_e^0}{c_e^i}\}$ denote the maximum cost inflation factor. We show that even when λ is polynomial in $n = |V(G)|$ the problem is hard to approximate.

Theorem 2. *Stochastic Spanning Tree problem on a graph $G = (V, E)$ with $n = |V(G)|$ with inflation λ is $O(\log n)$-hard to approximate unless $P = NP$ even when $\lambda = poly(n)$.*

We then give an approximation algorithm matching the lower bound proved in Theorem 2.

Theorem 3. *Given an instance of Stochastic Spanning Tree problem in the Black-box model with inflation factor λ, there exists a randomized algorithm which returns a solution of expected cost $O((\log n + \log \lambda) \cdot \mathrm{OPT})$ with high probability, where* OPT *is the cost of the optimum solution. The running time of the algorithm is polynomial in n and λ.*

Hence, if $\lambda = p(n)$ for some polynomial $p(n)$, we get an approximation algorithm optimal up to a constant factor. We reduce the Stochastic MST problem in the black-box model to the Stochastic MST with explicit scenarios by sampling $k = poly(n, \lambda)$ times and using these scenarios in the new problem constructed. We show that solving the Stochastic MST problem with these explicit scenarios gives a good solution to Stochastic MST problem in the Black-box model as well.

2.3 Stochastic MST with Independent Random Costs

A well-studied problem is to determine the cost of the minimum spanning tree of a complete graph when the edge costs are independent random variables uniformly distributed in $[0, 1]$. A classic result of Frieze [4] shows that the expected cost of the minimum spanning tree is $\zeta(3) + o(1)$. A natural stochastic version of the problem can be formulated when first-stage costs are given by a cost function c^0 and second-stage costs are independent random variables uniformly distributed between $[0, 1]$.

For this model, we analyze a thresholding heuristic. The thresholding heuristic with threshold α consists of excluding all edges with cost more than α in the first stage and constructing a minimum spanning tree over each of the resulting components. These components are then optimally joined in the second stage. We show that a thresholding heuristic gives a solution within $\frac{\zeta(3)}{4} + o(1)$ of the

optimum solution. We also show that if the optimum solution is 'tiny', then our solution is also small by showing that the expected cost of our solution is at most $\sqrt{3\zeta(3) \cdot \mathrm{OPT}} + o(1)$.

Theorem 4. *The thresholding heuristic with the threshold* $\frac{\zeta(3)}{n}$ *gives a solution whose expected cost is at most* $\mathrm{OPT} + \frac{\zeta(3)}{4} + o(1)$*, where* OPT *is the expected cost of the optimal solution. The expected cost is also bounded by* $\sqrt{3\zeta(3) \cdot \mathrm{OPT}} + o(1)$.

Another stochastic version of the problem, when both the first stage and second stage costs are independent random variables uniformly distributed between $[0, 1]$, was studied by Flaxman et al[3]. Flaxman et al[3] show that a thresholding heuristic gives a solution of cost $\zeta(3) - \frac{1}{2} + o(1)$, while $\frac{\zeta(3)}{2} + c$ for some $c > 0$ is a lower bound for the optimum. Surprisingly, the best threshold for their problem is $\frac{1}{n}$.

3 Stochastic MST with Explicit Scenarios

Linear Programming Formulation. The LP formulation for the two-stage Stochastic MST is the standard formulation enforcing the requirement that every non-trivial cut must be covered by an edge chosen in the first stage or, in each scenario, it must be covered with an edge chosen in the second stage. Although this LP formulation has an integrality gap of 2 in the deterministic setting, for stochastic version of the MST problem the optimal solution can be rounded to obtain a near optimal approximation algorithm. We call the following linear program the cut-cover LP.

$$
\begin{array}{lll}
 & \min \ \sum_e c_e^0 x_e^0 \ + \ \sum_e \sum_{i=1}^{k} p^i \, c_e^i x_e^i & \\
s.t. & & \\
 & \sum_{e \in \delta(S)} x_e^0 + x_e^i \ \geq \ 1 & \forall S \subset V, \ 1 \leq i \leq k \\
 & x_e^i \geq 0 & \forall e \in E, \ 0 \leq i \leq k
\end{array}
$$

We solve the the cut-cover linear programming formulation of the 2-stage spanning tree problem. Although, the linear program is not compact, an efficient separation subroutine can be supplied via an invocation to a min-cut subproblem. Alternately, equivalent compact reformulations can be readily obtained. (For eg. see the survey on spanning tree formulations by Magnanti and Wolsey [11]).

Rounding the Linear Program. Let $\hat{x}$ denote the optimal solution. We construct k forests, one for each scenario, by rounding the LP solution $\hat{x}$ in phases. Initially, all k forests have singleton components. In each phase, we pick edge e independently with probability $\hat{x}_e^0$ and include it in each of the k forests. Also for each i, $1 \leq i \leq k$, we pick edge e independently with $\hat{x}_e^i$ and include it in the i^{th} forest. Clearly, the expected cost of edges included in each phase is precisely OPT where OPT is the cost of the optimal LP solution $\hat{x}$. We argue that in one phase, for each of the k forests, the number of components decrease by a factor

of $\frac{9}{10}$ with probability at least $\frac{1}{2}$. We then argue that in $O(\log n + \log k)$ phases, with very high probability, each of the k forests will have just one component, i.e., a feasible solution for the 2-stage spanning tree problem and the expected cost of the solution is $O(\log n + \log k) \cdot \mathrm{OPT}$ as claimed. We elaborate on the above outline.

Lemma 1. *The expected cost of the edges paid in any phase is at most* OPT.

Proof. An edge e is included in the first stage with probability $\hat{x}_e^0$ and in the i^{th} scenario with probability $\hat{x}_e^i$. Hence, the expected cost paid for including this edge is $c_e^0 \hat{x}_e^0 + \sum_{i=1}^k p^i c_e^i \hat{x}_e^i$. Hence, the total expected cost paid in any phase is $\sum_{e \in E}(c_e^0 \hat{x}_e^0 + \sum_{i=1}^k p^i c_e^i \hat{x}_e^i)$ which is exactly OPT. □

Let F_j^i denote the i^{th} forest after j phases. Let C_j^i denote the number of components in F_j^i. We call a phase j 'successful' for scenario i, if $C_j^i < 0.9 C_{j-1}^i$ or if $C_{j-1}^i = 1$. We now state a lemma from [1]. For completeness, we also include the proof in the Appendix.

Lemma 2 (Alon [1]). *For every i, j, the conditional probability that phase j is successful for scenario i, given any set of components in F_{j-1}^i, is at least $\frac{1}{2}$.*

Lemma 3. *After $t = (40 \log n + 16 \log k)$ phases, the probability that any of the k forests, F_t^i, $1 \le i \le k$, is not connected is at most $\frac{1}{(kn)^2}$.*

Proof. Observe that the i^{th} forest gets connected after at most $\log_{0.9} n < 10 \log n$ successful phases. Hence, if F_t^i is not connected, then there have been at most $10 \log n$ successful phases for the i^{th} scenario. The probability of this event is no more than the probability that we get at most $10 \log n$ heads in t independent tosses of a fair coin. Although, the event that a phase j is successful for scenario i is not independent for different j, we can apply the above bound since Lemma 2 gives a lower bound on success given any history. Using estimates for tails of Binomial distributions, this probability is at most

$$e^{-(10 \log n + 8 \log k)^2 / 2 \times (20 \log n + 8 \log k)} \le e^{-(10 \log n + 8 \log k)/4} \le \frac{1}{(kn)^2}.$$

Using union bound, we get that the probability that any of the k forests is not connected is at most $k \times \frac{1}{(kn)^2} = \frac{1}{kn^2}$. □

Hence, after $40 \log n + \log k$ phases with very high probability each of the k forests are connected and from Lemma 1, the expected cost of the solution is $(40 \log n + \log k) \cdot \mathrm{OPT}$ proving Theorem 1.

Remark 1. If the randomized rounding fails to connect any forest after $O(\log n + \log k)$ phases, then we use a simple k-approximation algorithm: for each scenario build a tree by running Kruskal's algorithm on edge costs equal to the minimum of first and second stage costs. This will guarantee that our algorithm always *builds a spanning tree; further, since the failure probability is at most $\frac{1}{(kn)^2}$ per scenario the expected cost of our solution is still bounded by $O(\log n + \log k)$ times the optimum.*

Remark 2. Theorem 1 shows that the cut-cover LP formulation has a integrality gap of $O(\log n)$ assuming $k = poly(n)$. Also, the reduction from set cover given by Gupta [5], yields examples where the cut-cover LP has a integrality gap of $\Omega(\log n)$. Hence, the integrality gap of the cut-cover LP formulation is $\theta(\log n)$.

4 Approximation Algorithm for Black-Box Model

In this section, we consider the black-box model for the Stochastic spanning tree problem. We assume that the inflation factor λ is bounded by some polynomial $p(n)$. Note that the problem still remains $O(\log n)$-hard to approximate unless P = NP (Theorem 2).

Let TRUE-LP denote the linear programming formulation of the stochastic spanning tree problem. Given any $\epsilon > 0, \delta > 0$, we sample from the black box, according to probability distribution ρ, the second stage scenarios $k = poly(n, \lambda, \frac{1}{\epsilon}, \frac{1}{\delta})$ times. We then formulate a new stochastic MST problem using these k samples as the second stage scenarios each occurring with probability $\frac{1}{k}$. We call the corresponding linear program of the new instance of the problem formed SAMPLE-LP.

Let $\hat{x}^0$ denote the first stage component of the optimal solution of SAMPLE-LP. We show that if $\hat{x}^0$ is used as the first stage solution for TRUE-LP, with probability $1 - \delta$, we can extend this solution in any scenario with total expected cost $(1 + \epsilon)\,\mathrm{OPT}$ where is the expected cost of the optimal solution of TRUE-LP. Hence, using the result of Theorem 1, we round this *LP* solution to get an integral solution of expected cost $O(\log n \cdot \mathrm{OPT})$. Observe that, to obtain the first stage solution we only need to know the first stage fractional variables $\hat{x}^0$ and need not know the second stage distribution or second stage solution.

Given any x^0, the first stage variables, let $f(x^0)$ denote the cost of extending x^0 to a feasible solution on a random sample according to probability distribution ρ and let $F(x^0) = \mathbf{E}\big[f(x^0)\big]$. Let $F_k(x^0)$ be the random variable denoting the average completion cost over k independently sampled scenarios, i.e., $F_k(x^0) = \frac{1}{k}\sum_{i=1}^{k} f_i(x^0)$ where $f_i(x^0)$ is the random variable denoting the cost of extending x^0 in the i^{th} sample. The key to bounding our sample size is Lemma 4. The techniques of [15, 14] also can also be used to prove Lemma 4, we here give a much simpler proof. For simplicity, we assume that probability distribution ρ is discrete and finite but the same argument also works for continuous distributions.

Lemma 4. *Given any $\epsilon > 0$ and $\delta > 0$, if $k = \frac{\lambda^4}{\epsilon^2\delta}$, then $Pr[|F_k(x^0) - F(x^0)| < \epsilon \cdot \mathrm{OPT}] \leq \delta$. Here,* OPT *is the expected cost of the optimal solution.*

Proof. Clearly $\mathbf{E}\big[F_k(x^0)\big] = \frac{1}{k}\sum_{i=1}^{k} \mathbf{E}\big[f_i(x^0)\big] = \frac{1}{k}\sum_{i=1}^{k} F(x^0) = F(x^0)$. Also,

$$Var[F_k(x^0)] = \frac{1}{k^2}\sum_{i=1}^{k} Var[f_i(x^0)] = \frac{1}{k} Var[f(x^0)] \tag{1}$$

as $f^i(x^0)$ are independent random variables with distribution identical to $f(x^0)$.

Now,

$$Var[f(x^0)] = \sum_t \rho^t (c^t(x^0) - F(x^0))^2 \tag{2}$$

where ρ^t is the probability of the t^{th} scenario and $c^t(x^0)$ is the cost paid in this scenario to extend x^0 to a feasible solution. Let j denote the index for which $c^t(x^0)$ is minimum. As we can select exactly the edges selected in the j^{th} scenario in any other scenario to extend x^0 and inflation factor is bounded by λ, hence

$$c^t(x^0) \leq \lambda^2 c^j(x^0) \tag{3}$$

Also, if we do not select anything in the first stage our cost only goes up by a factor of λ and hence $\sum_t \rho^t c^t(x^0) \leq \lambda \cdot \mathrm{OPT}$. As $c^j(x^0)$ is minimum, this implies

$$c^j(x^0) \leq \lambda \cdot \mathrm{OPT} \tag{4}$$

From Equation (2), (3) and (4), we have

$$\begin{array}{rcl} Var[f(x^0)] & \leq & \sum_t \rho^t c^t(x^0)^2 \\ & \leq & (\lambda^2 c^j(x^0)) \sum_t \rho^t c^t(x^0) \\ & = & \lambda^2 c^j(x^0) \cdot \lambda \cdot \mathrm{OPT} \\ & \leq & \lambda^4 \, \mathrm{OPT}^2 \end{array} \tag{5}$$

Hence, using Equation (1),

$$Var[F_k(x^0)] = \frac{1}{k} Var[f(x^0)] = \epsilon^2 \delta \cdot \mathrm{OPT}^2 \tag{6}$$

Hence, by the Chebychev inequality we have,

$$Pr[|F_k(x^0) - F(x^0)| > \epsilon \cdot \mathrm{OPT}] \leq \frac{\epsilon^2 \delta \cdot \mathrm{OPT}^2}{\epsilon^2 \cdot \mathrm{OPT}^2} = \delta \tag{7}$$

□

Proof of Theorem 3. Let $\hat{x}^0$ denote the first stage component of the optimial solution of SAMPLE-LP formed using $k = \frac{\lambda^4}{\epsilon^2 \delta}$ samples. Also let $\bar{x}^0$ denote the first stage optimal solution to TRUE-LP. Using Lemma 4 twice, once for $\hat{x}^0$ and once for $\bar{x}^0$, we have with probability at least $1 - 2\delta$ that

$$|F_k(\hat{x}^0) - F(\hat{x}^0)| \leq \epsilon \cdot \mathrm{OPT}$$
$$|F_k(\bar{x}^0) - F(\bar{x}^0)| \leq \epsilon \cdot \mathrm{OPT}$$

As $\hat{x}^0$ is the optimal solution to SAMPLE-LP and $\bar{x}^0$ is the optimal solution to TRUE-LP, we have

$$F_k(\hat{x}^0) \leq F_k(\bar{x}^0),\ F(\bar{x}^0) \leq F(\hat{x}^0)$$

Using the above we have that

$$F(\hat{x}^0) \leq F(\bar{x}^0) + 2\epsilon \cdot \mathrm{OPT} \tag{8}$$

Hence, sampling $k = \frac{\lambda^4}{\epsilon^2\delta}$ times and solving the SAMPLE-LP gives first stage variables which can be extended to second stage with total cost at most $(1+\epsilon)\,\mathrm{OPT}$ with probability at least $1-2\delta$. Hence, now if we apply Theorem 1 to SAMPLE-LP, we obtain the first stage component of a solution of total cost $O((\log n + \log\lambda + \log\frac{1}{\epsilon\delta})\,\mathrm{OPT})$ with probability at least $1-\delta$ proving Theorem 3. □

5 Stochastic MST with Random Costs

In this section, we consider the two stage stochastic MST when the second stage costs are given by independent random variables uniformly distributed between $[0,1]$.

Thresholding. A thresholding heuristic with threshold α consists of removing all edges in the first stage with costs more than α and constructing a minimum cost spanning tree over each of the components formed. The components are then joined in an optimal manner in the second stage.

Observe that, for a forest F, the cost of extending F to a spanning tree in the second stage depends only on the sizes of the components of F. Let the components of F be $(C_1,\ldots,C_k)$. Construct an auxillary graph F' with k vertices v_i, $1 \le i \le k$ where v_i corresponds to component C_i of F. Include $|C_i|\cdot|C_j|$ edges between v_i and v_j, where the cost of edges are independent random variables distributed uniformly between $[0,1]$. Clearly, the cost of extending F to a spanning tree in the second stage is the cost of minimum spanning tree of F' which only depends on the sizes of the components of F.

In Lemma 6, we show that if we join k components then the expected average cost of each edge bought tomorrow is between $(\frac{k}{n})\frac{\zeta(3)}{n} + o(\frac{1}{n})$ and $\frac{\zeta(3)}{n} + o(\frac{1}{n})$. Hence, it is 'reasonable' not to buy any edge which costs more than $\frac{\zeta(3)}{n}$ and buy the edges which cost less than $\frac{\zeta(3)}{n}$ in the first stage itself.

First, we prove a lemma which shows how the cost of the second stage depends on the sizes of the components formed in the first stage. Note that the cost of extending a forest F to a spanning tree in the second is the exactly the cost of minimum cost spanning tree in the auxillary graph F' defined above.

Lemma 5. *Let G and G' be two forests, where the components of G are $(C_1, C_2, \ldots, C_k)$ and the components of G' are $(C_1\cup\{x\}, C_2\setminus\{x\}, C_3,\ldots,C_k)$ for some vertex $x \in C_2$. If $|C_1| \ge |C_2|$, then the expected cost of extending G to a spanning tree in the second stage is less than the cost of extending G' to a spanning tree.*

Proof. Let $\mathbf{E}[c(G)]$ denote the expected cost of extending G to a spanning tree in the second stage. Consider graph H which has components $(C_1, C_2', x, C_3,\ldots,C_k)$, where $C_2' = C_2 \setminus x$. If we connect up x to C_1 for free, then we get G', and if we connect x to C_2' for free, then we get G. Therefore,

$$\mathbf{E}[c(H)] \ge \mathbf{E}[c(G)] \qquad \text{and} \qquad \mathbf{E}[c(H)] \ge \mathbf{E}[c(G')]$$

We will show that the reduction in cost when connecting x to C_1 is larger. This will imply that $\mathbf{E}[c(G)] \leq \mathbf{E}[c(G')]$.

Let G_p denote the graph formed by including in G every edge independently with probability p. Similarly, define G'_p and H_p. In order to bound expected cost of spanning trees on G and G', we look at the expected number of connected components in G_p, G'_p and H_p. Let $\chi(G_p)$ and $\chi(G'_p)$ denote the number of connected components of G_p and G'_p respectively. We will show that $\mathbf{E}[\chi(G_p)] \leq \mathbf{E}\big[\chi(G'_p)\big]$. Note that this suffices in proving the above claim, since $\mathbf{E}[c(G)] = \int_0^1 (\mathbf{E}[\chi(G_p)] - 1)\, dp$.

Note that we can obtain G_p (resp. G'_p) from H_p by connecting x to C_1 (resp. C'_2) in H_p. We focus only on the number of connected components.

We use the *principle of deferred decision*. We first reveal the edges coming out of vertex x in H_p. If x has edges to both C_1 and C'_2 or to neither of them, then adding an edge from x to either C_1 or C'_2 and revealing rest of the edges in the graph gives the expected number of connected components.

Now observe that since $|C_1| > |C'_2|$, the probability that x is connected to component C_1 in H_p is reater than the probability that x is connected to C'_2. Therefore, G'_p formed by including x to C'_2 in H_p, has higher probability of reducing the number of connected components by 1 than in G_p, formed by including x in C_1. From this it follows that $\mathbf{E}[\chi(G_p)] \leq \mathbf{E}\big[\chi(G'_p)\big]$. This proves the lemma. □

The cost of extending a forest with k components to a spanning tree in the second stage can now be bounded as follows. The proof involving some careful calculations appears in the Appendix.

Lemma 6. *Given a forest F with k components, the expected cost of extending F to a spanning tree in the second stage is between $(\frac{k}{n})^2\zeta(3)+o(1)$ and $\frac{k}{n}\zeta(3)+o(1)$.*

Using this lemma, we now prove the main result of this section.

Proof of Theorem 4. Now, we prove Theorem 4 which bounds the expected cost of the solution returned by a thresholding heuristic. First, observe that the optimum solution must buy a forest in the first stage and join the components of the forest in the second stage. Suppose, the thresholding heuristic and the optimum solution buys k and opt_1 edges respectively, in the first stage. Then, they buy $n-k-1$ and $n-opt_1-1$ edges respectively, in the second stage. We form two cases depending on which of k and opt_1 is larger.

Case I ($k \leq opt_1$). Let the cost of the k edges bought by the thresholding heuristic be $c_1(k)$. List the opt_1 edges included in the optimum solution in the first stage in increasing costs. Clearly, cost of the first k edges in the order is at least $c_1(k)$ as we chose the cheapest k acyclic edges. Also, rest of the $opt_1 - k$ edges in the sequence must cost at least $\frac{\zeta(3)}{n}$ as any acyclic subgraph of edges with cost at most $\frac{\zeta(3)}{n}$ is of size at most k. If $\mathbf{E}[c(G)]$ denotes the expected cost of the thresholding heuristic and *Loss* denotes $\mathbf{E}[c(G)] - \mathrm{OPT}$,then by Lemma 6 we have,

$$\mathbf{E}[c(G)] \leq c_1(k) + \left(\frac{n-k-1}{n}\right)\zeta(3) + o(1)$$

$$\text{and} \quad \text{OPT} \geq c_1(k) + (opt_1 - k)\frac{\zeta(3)}{n} + \left(\frac{n-opt_1-1}{n}\right)^2\zeta(3) - o(1)$$

$$\Rightarrow \quad Loss \leq \left(\frac{n-opt_1-1}{n}\right)\zeta(3) - \left(\frac{n-opt_1-1}{n}\right)^2\zeta(3) + o(1)$$

Case II ($k > opt_1$). Let the cost of the opt_1 edges bought by the optimum solution be $c_1(opt_1)$. Order the k edges bought by the thresholding heuristic in increasing order of costs. The cost of the first opt_1 edges in this order is at most $c_1(opt_1)$ as we buy the cheapest acyclic subgraph of size opt_1. Also, each of the last $k - opt_1$ edges in the order costs at most $\frac{\zeta(3)}{n}$ as we threshold at $\frac{\zeta(3)}{n}$. Hence, using Lemma 6, we get

$$\mathbf{E}[c(G)] \leq c_1(opt_1) + (k - opt_1)\frac{\zeta(3)}{n} + \left(\frac{n-k-1}{n}\right)\zeta(3) + o(1)$$

$$\text{and} \quad \text{OPT} \geq c_1(opt_1) + \left(\frac{n-opt_1-1}{n}\right)^2\zeta(3) - o(1)$$

$$\Rightarrow \quad Loss \leq \left(\frac{n-opt_1-1}{n}\right)\zeta(3) - \left(\frac{n-opt_1-1}{n}\right)^2\zeta(3) + o(1)$$

Now, $(\frac{n-opt_1-1}{n})\zeta(3) - (\frac{n-opt_1-1}{n})^2\zeta(3) \leq \frac{\zeta(3)}{4}$ for any value of opt_1. Hence, we get that

$$\mathbf{E}[c(G)] \leq \text{OPT} + \frac{\zeta(3)}{4} + o(1)$$

for both of the cases, proving the first claim in the theorem.

Now observe that $\text{OPT} \leq \zeta(3)$ and consequently, each of the terms in the expression of OPT must be less than $\zeta(3)$. Using this, in case I, we have that

$$\begin{aligned}
\mathbf{E}[c(G)] &\leq c_1(k) + \left(\frac{n-k-1}{n}\right)\zeta(3) + o(1) \\
&= c_1(k) + \left(\frac{opt_1-k}{n}\right)\zeta(3) + \left(\frac{n-opt_1-1}{n}\right)\zeta(3) + o(1) \\
&\leq \sqrt{3\left(c_1(k)^2 + \left(\left(\frac{opt_1-k}{n}\right)\zeta(3)\right)^2 + \left(\frac{n-opt_1-1}{n}\right)\zeta(3)\right)^2} + o(1) \\
&\leq \sqrt{3(\zeta(3)c_1(k) + \zeta(3)(opt_1-k)\frac{\zeta(3)}{n} + \zeta(3)(\frac{n-opt_1-1}{n})^2\zeta(3))} + o(1) \\
&\leq \sqrt{3\zeta(3)\cdot\text{OPT}} + o(1)
\end{aligned}$$

The second step of the inequality follows from the fact that $\sqrt{3(a^2+b^2+c^2)} \geq a+b+c$ for any reals a, b, c.

In the second case, we have that

$$\begin{aligned}
\mathbf{E}[c(G)] &\leq c_1(k) + \left(\frac{k - opt_1}{n}\right)\zeta(3) + \left(\frac{n - opt_1 - 1}{n}\right)\zeta(3) \ + \ o(1) \\
&= c_1(opt_1) + \left(\frac{n - opt_1 - 1}{n}\right)\zeta(3) \ + \ o(1) \\
&\leq \sqrt{2\left(c_1(opt_1)^2 + \left(\left(\frac{n - opt_1 - 1}{n}\right)\zeta(3)\right)^2\right)} \ + \ o(1) \\
&\leq \sqrt{2\left(\zeta(3)c_1(opt_1) + \zeta(3)\left(\frac{n - opt_1 - 1}{n}\right)^2\zeta(3)\right)} \ + \ o(1) \\
&\leq \sqrt{2\zeta(3)\cdot \mathrm{OPT}} \ + \ o(1)
\end{aligned}$$

Here, the second step follows from $\sqrt{2(a^2 + b^2)} \geq a + b$ for any reals a, b. Hence, $c(T) \leq \sqrt{3\zeta(3)\cdot \mathrm{OPT}} + o(1)$ as claimed.

References

1. N. Alon, *A note on network reliability,* Discrete Probability and Algorithms, D. Aldous et al., eds., IMA Volumes in mathematics and its applications, Vol. 72, Springer Verlag (1995), 11-14.
2. J. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer, 1997.
3. A D Flaxman, A M Frieze and M Krivelevich. *On the random 2-stage minimum spanning tree.* Preprint (2004).
4. A M Frieze. On the value of a random minimum spanning tree problem, *Discrete Applied Mathematics 10 (1985) 47-56.*
5. Anupam Gupta. *Personal Communication (2004).*
6. Anupam Gupta, R. Ravi, Amitabh Sinha. *Boosted Sampling: Approximation Algorithms for Stochastic Optimization.* In Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, 2004.
7. Anupam Gupta, R. Ravi, Amitabh Sinha. *An edge in time saves nine: LP rounding Approximation Algorithms for Stochastic Network Design.* In the Proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science, 2004.
8. U. Feige. *A threshold of* ln *for approximating set cover.* J. ACM 45, 634-652,1998.
9. Nicole Immorlica, David Karger, Maria Minkoff, Vahab S. Mirrokni. *On the Costs and Benefits of Procastination: Approximation Algorithms for Stochastic Combinatorial Optimization Problems*, Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms. Pages: 691 - 700, 2004.
10. David Karger, Philip Klein, Robert Tarjan. *A randomized linear-time algorithm to find minimum spanning trees*, Journal of the ACM, Volume 42, Issue 2, Pages: 321 - 328, 1995.
11. T. L. Magnanti, L. A. Wolsey. *Optimal Trees, Handbook in OR and MS*, Volume 7, Eds M.O. Ball et al, 503-615, 1995.
12. R. Ravi, Amitabh Sinha. *Hedging Uncertainty: Approximation Algorithms for Stochastic Optimization Problems.* In Proceedings of the 10th International Conference on Integer Programming and Combinatorial Optimization (IPCO), 2004.

13. Ran Raz, S. Safra. *A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP.* Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, pages 475-484, 1999.
14. A. Shapiro. *Monte Carlo sampling approach to stochastic programming.* Proceedings of 2003 MODE-SMAI Conference Pau, France,pages 65-73, March 27-29, 2003
15. David B. Shmoys, Chaitanya Swamy. *Stochastic Optimization is (Almost) as easy as Deterministic Optimization* Proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04), October 17 - 19, 2004 Rome, Italy
16. M. H. van der Vlerk. *Stochastic programming bibliography.* World Wide Web, http://mally.eco.rug.nl/spbib.html, 1996-2003.

Appendix A: Hardness of Stochastic MST with Bounded Inflation

In this section, we prove Theorem 2 stating that Stochastic MST with polynomial inflation is $O(\log n)$-hard to approximate unless $NP \subseteq \text{DTIME}(n^{O(\log \log n)})$. We give an approximation preserving transformation from the Set Cover problem to the Stochastic MST with inflation $\lambda = p(n)$ for some polynomial $p(n)$.

Given an instance of set cover problem, i.e, a set $U = [n]$ and collection of subsets $S_1, \ldots, S_m$, we construct an instance of stochastic spanning tree problem with $m+n+1$ vertices. Let the vertices be $\{v_1, \ldots, v_n, s_1, \ldots, s_m, r\}$. The vertex v_i corresponds to element $i \in U$ and s_j corresponds to subset S_j.

The first stage cost of edge (r, s_j) is 1, for each $1 \leq j \leq m$ and cost of all other edges is mn. See Figure 1(a). The edges not present have cost mn.

There are n scenarios, one corresponding to each element $i \in U$, each having a probability $\frac{1}{n}$ of appearance in the second stage. In the i^{th} scenario, each edge of cut separating $T_i = \{v_i\} \cup \{s_j : i \in S_j\}$ from $G \setminus T_j$ has cost mn and remaining edges have cost $\frac{1}{mn}$. See Figure 1(b). The edges absent have cost mn.

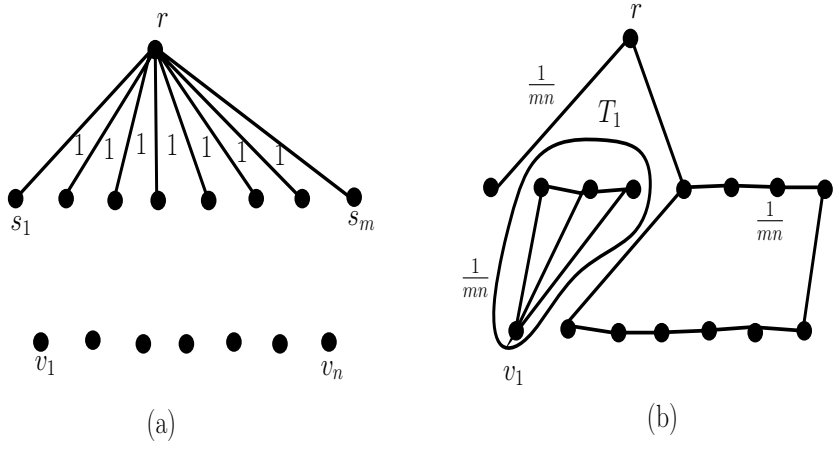


Fig. 1. (a) First stage costs (b) Second stage scenario corresponding to $_1$

It is clear that the inflation factor of any edge is bounded by m^2n^2. Given any set cover C of size k, choose edges (r, v_j) such that $S_j \in C$ in the first stage. In the i^{th} scenario, this first stage solution can be extended to a spanning tree with edges of cost $\frac{1}{mn}$. Hence, there exists a feasible solution of stochastic spanning tree of total cost at most $k + \frac{1}{n}$. Given a feasible solution to the stochastic spanning tree of total cost $c \leq m + 1$, we construct a feasible set cover of cost between $c - \frac{1}{n}$ and c. Clearly, the solution contains no edge of cost mn. Let $C = \{S_j : (r, s_j)$ is selected in the first stage$\}$. In the i^{th} scenario, we cannot buy any edge of cost mn. Hence, the i^{th} element must have been covered by some set in C. Hence, the sets selected form a set cover. Also, the total cost paid in the second stage is at most $\frac{1}{n}$. Hence, the size of C is at least $c - \frac{1}{n}$. The cost of the set cover is exactly the first stage cost paid in the solution.

Hence, the hardness of the approximation result for the set cover [8, 13] imply that the stochastic spanning tree is at least $O(\log n)$-hard to approximate unless $P = NP$.

Appendix B: Proofs

Proof of Lemma 2. If F^i_{j-1} is connected, then the claim is trivial. Given some forest F^i_{j-1}, shrink each component to a singleton vertex and call the constructed multi-graph H. Consider any vertex v in H. An edge e is not included in F^i_j with probability $(1-x^0_e)(1-x^i_e)$. Let $\delta(v)$ denote the neighborhood of node v. Hence, the probability that v remains isolated is

$$\prod_{e\in\delta(v)} (1 - x^0_e)(1 - x^i_e) \leq exp(- \sum_{e\in\delta(v)} (x^0_e + x^i_e))$$

Using the fact that $\sum_{e\in\delta(v)}(x^0_e + x^i_e) \geq 1$, we get that the probability that v is isolated is at most $\frac{1}{e}$. Using linearity of expectation, the expected number of isolated vertices in H is $|H|/e$, and hence with probability at least $\frac{1}{2}$, the number of isolated vertices is less than $2|H|/e$. Hence, the number of connected components in F^i_j is at most

$$\frac{2|H|}{e} + \frac{1}{2}(|H| - 2|H|/e) = (\frac{1}{2} + \frac{1}{e})|H| < 0.9|H|$$

Using that $|H| = C^i_{j-1}$, we get the desired result.

Proof of Lemma 6. Let $\mathbf{E}[c(F)]$ denote the expected cost of extending the forest F to a spanning tree. By Lemma 5, we get that $\mathbf{E}[c(F)]$ is minimum when the k components of F are of equal sizes and it is maximum when all except one component are singletons. We calculate the expected costs of joining k components in either cases and get appropriate bounds as claimed.

Consider the case when F has one component of size $n - k + 1$ and rest $k - 1$ components are singleton vertices. Then, the expected cost of joining the components optimally is

$$\mathbf{E}[c(F)] = \int_{p=0}^{1} \mathbf{E}[\kappa(F_p) - 1]\, dp \tag{9}$$

where F_p is the graph formed when each edge is included in F, independently, with probability p and $\kappa(F_p)$ denotes the connected component of F_p. Now, using the ideas from [3], we need to consider components of size at most $(\log n)^2$. There are at most $\frac{n}{(\log n)^2}$ components of size greater than $(\log n)^2$ and each can be joined with probability $1 - o(1)$ with an edge of cost at most $\frac{\log n}{n}$. Hence, the total cost of joining such components is $o(1)$. Also, we can assume that $n - k + 1 \geq (\log n)^2$, otherwise the claim holds trivially as otherwise $\frac{k-1}{n}\zeta(3) + o(1) \geq \frac{n-(\log n)^2}{n}\zeta(3) + o(1) = \zeta(3) + o(1)$. Hence, we assume that the that the largest component of F is not included in a component of size more than $(\log n)^2$. Using the above observations we have that,

$$\begin{aligned}
\mathbf{E}[c(F)] &= \int_{p=0}^{1} \sum_{j=1}^{(\log n)^2} \binom{k-1}{j} j^{j-2}\, p^{j-1}(1-p)^{j(n-j)+O(j^2)} dp \;+\; o(1) \\
&= \sum_{j=1}^{(\log n)^2} \binom{k-1}{j} j^{j-2} \int_{p=0}^{1} p^{j-1}(1-p)^{jn}\, dp \;+\; o(1) \\
&= \sum_{j=1}^{(\log n)^2} \binom{k-1}{j} j^{j-2} \frac{(j-1)!(jn)!}{(j+jn)!} + o(1) \\
&= \sum_{j=1}^{(\log n)^2} \frac{(k-1)^j}{j!} j^{j-2} \frac{(j-1)!}{j^j n^j} + o(1) \\
&= \sum_{j=1}^{(\log n)^2} \left(\frac{k-1}{n}\right)^j \frac{1}{j^3} + o(1) \\
&\leq \frac{k-1}{n} \sum_{j=1}^{(\log n)^2} \frac{1}{j^3} + o(1) = \frac{k}{n}\zeta(3) + o(1)
\end{aligned}$$

Now consider the case when all k components are nearly of equal sizes. Assume for the sake of simplicity that all the components are of size $\frac{n}{k}$. Then F is equivalent to a multi-graph on k vertices, each vertex representing a component in F. The multi-graph has $(\frac{n}{k})^2$ edges between any two vertices. We replace the $(\frac{n}{k})^2$ edges edges with a single edge whose cost is a random variable defined to be the minimum of $(\frac{n}{k})^2$ independent random variables uniformly distributed between $[0, 1]$. Observe that the new random variable has is distributed around the origin like a uniform random variable between $[0, (\frac{k}{n})^2]$. Hence, using the result by Frieze [4], the cost of the minimum cost spanning tree is $(\frac{k}{n})^2\zeta(3)$. The argument goes through even in the case when k does not divide n and all components differ in sizes by at most one and the same bound holds. The calculations are tedious and are omitted.

Combinatorial Analysis of Generic Matrix Pencils

Satoru Iwata and Ryo Shimizu

University of Tokyo, Tokyo 113-8656, Japan
iwata@mist.i.u-tokyo.ac.jp
http://www.sr3.t.u-tokyo.ac.jp

Abstract. This paper investigates the Kronecker canonical form of matrix pencils under the genericity assumption that the set of nonzero entries is algebraically independent. We provide a combinatorial characterization of the sums of the row/column indices supported by efficient bipartite matching algorithms. We also give a simple alternative proof for a theorem of Poljak on the generic ranks of matrix powers.

1 Introduction

A matrix pencil is a polynomial matrix whose nonzero entries are of degree at most one. Based on the theory of elementary divisors, Weierstrass established a criterion for strict equivalence, as well as a canonical form, of regular matrix pencils. Somewhat later, Kronecker investigated singular pencils to obtain a canonical form for matrix pencils in general under strict equivalence transformations, which is now called the Kronecker canonical form.

The Kronecker canonical form finds a variety of applications in control theory of linear dynamical systems [2, 18]. It is also in a close relation to the index of differential algebraic equations [7, 8, 16].

Numerically stable algorithms are already available for computing the Kronecker canonical form [1, 3, 11, 20, 21]. Nevertheless, these algorithms are not very accurate in the presence of round-off errors. The numerical difficulty is inherent in the problem as the Kronecker canonical form is highly sensitive to perturbation. This recently motivates extensive research on perturbation of matrix pencils [5, 6].

On the other side, matrix pencils arising in applications are often very sparse and their entries that represent physical characteristics are not precise in value because of noises. Hence one may assume that there is no algebraic dependency among the nonzero entries. It is then desirable to predict the structure of the Kronecker canonical form efficiently from the combinatorial information such as the zero/nonzero pattern without numerical computation. Such a structural approach has been conducted for regular matrix pencils by Duff and Gear [4] and Pantelides [16] in the context of differential algebraic equations. Murota [13] described a complete characterization of the Kronecker canonical form of regular matrix pencils in terms of the maximum degree of minors, which is tantamount

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 335–348, 2005.

to the maximum weight of bipartite matchings under the genericity assumption. An extension of this characterization to mixed matrix pencils is also presented in [14, 15]. A recent paper of van der Woude [19] provides another combinatorial characterization based on the Smith normal form [12].

In this paper, we extend the structural approach to the analysis of singular matrix pencils. The possible existence of the minimal row/column indices (rectangular blocks in the canonical form) makes this problem much more complicated than the regular case. In fact, it remains open to design an efficient algorithm for determining the row/column indices. The main result of this paper, however, is an efficiently computable combinatorial characterization of the sums of the minimal row/column indices using the Dulmage-Mendelsohn decomposition and weighted bipartite matchings. The sum of the indices provides significant information such as the controllability of the linear dynamical system described by the matrix pencil.

The structure of the Kronecker canonical form is closely related to the series of larger constant matrices, called the expanded matrices. We investigate those expanded matrices to show that their ranks are equal to the term-ranks in most cases. As a byproduct, we give a simple alternative proof for a theorem of Poljak [17] on the generic ranks of matrix powers.

The outline of this paper is as follows. Section 2 recapitulates the Kronecker canonical form. In Section 3, we introduce generic matrix pencils. In Section 4, we explain the Dulmage-Mendelsohn decomposition, which plays an essential rôle in our result on the minimal row/column indices presented in Section 5. Sections 6 and 7 are devoted to the analysis of expanded matrices. The simple proof for the theorem of Poljak is shown in Section 8.

2 The Kronecker Canonical Form of Matrix Pencils

Let $D(s) = sA + B$ be an $m \times n$ matrix pencil of rank r with the row set R and the column set C. We denote by $D(s)[X, Y]$ the submatrix of $D(s)$ determined by $X \subseteq R$ and $Y \subseteq C$. A matrix pencil $\bar{D}(s)$ is said to be strictly equivalent to $D(s)$ if there exists a pair of nonsingular constant matrices U and V such that $\bar{D}(s) = UD(s)V$. A matrix pencil $D(s) = sA + B$ is said to be regular if $\det D(s) \neq 0$ as a polynomial in s. It is strictly regular if both A and B are nonsingular matrices.

For a positive integer μ, we consider $\mu \times \mu$ matrix pencils N_μ and K_μ defined by

$$N_\mu = \begin{pmatrix} 1 & s & 0 & \cdots & 0 \\ 0 & 1 & s & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1 & s \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}, \qquad K_\mu = \begin{pmatrix} s & 1 & 0 & \cdots & 0 \\ 0 & s & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & s & 1 \\ 0 & \cdots & \cdots & 0 & s \end{pmatrix}.$$

For a positive integer ε, we further denote by L_ε an $\varepsilon \times (\varepsilon+1)$ matrix pencil

$$L_\varepsilon = \begin{pmatrix} s & 1 & 0 & \cdots & 0 \\ 0 & s & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & s & 1 \end{pmatrix}.$$

We also denote by $L_\eta^\top$ the transpose matrix of L_η.

The following theorem establishes the Kronecker canonical form of matrix pencils under strict equivalence transformations.

Theorem 1 (Kronecker, Weierstrass). *For any matrix pencil $D(s)$, there exists a pair of nonsingular constant matrices U and V such that $\bar{D}(s) = UD(s)V$ is in a block-diagonal form*

$$\bar{D}(s) = \text{block-diag}(H_\nu, K_{\rho_1}, \cdots, K_{\rho_c}, N_{\mu_1}, \cdots, N_{\mu_d}, L_{\varepsilon_1}, \cdots, L_{\varepsilon_p}, L_{\eta_1}^\top, \cdots, L_{\eta_q}^\top, O),$$

where $\rho_1 \geq \cdots \geq \rho_c > 0$, $\mu_1 \geq \cdots \geq \mu_d > 0$, $\varepsilon_1 \geq \cdots \geq \varepsilon_p > 0$, $\eta_1 \geq \cdots \geq \eta_q > 0$, and H_ν is a strictly regular matrix pencil of size ν. The numbers c, d, p, q, ν, $\rho_1, \cdots, \rho_c$, $\mu_1, \ldots, \mu_d$, $\varepsilon_1, \cdots, \varepsilon_p$, $\eta_1, \cdots, \eta_q$ are uniquely determined. □

The block-diagonal matrix pencil $\bar{D}(s)$ in Theorem 1 is often referred to as the Kronecker canonical form of $D(s)$. The numbers $\mu_1, \ldots, \mu_d$ are called the indices of nilpotency. The numbers $\varepsilon_1, \cdots, \varepsilon_p$ and $\eta_1, \cdots, \eta_q$ are the minimal column and row indices, respectively. These numbers together with $\nu, \rho_1, \cdots, \rho_c$ are collectively called the structural indices of $D(s)$. Note that the matrix pencil $D(s)$ is regular if and only if $p = q = 0$.

For a polynomial $g(s)$ in s, let $\deg g(s)$ and $\operatorname{ord} g(s)$ denote the highest and lowest degrees of nonvanishing terms of $g(s)$, respectively. For each $k = 1, \ldots, r$, we denote

$$\delta_k(D) = \max\{\deg \det D(s)[X, Y] \mid |X| = |Y| = k, X \subseteq R, Y \subseteq C\},$$
$$\zeta_k(D) = \min\{\operatorname{ord} \det D(s)[X, Y] \mid |X| = |Y| = k, X \subseteq R, Y \subseteq C\}.$$

Note that $\delta_k(D)$ is concave in k and $\zeta_k(D)$ is convex in k. The following well-known lemma asserts that δ_k and ζ_k are invariant under strict equivalence transformations.

Lemma 1. *If $\bar{D}(s)$ is strictly equivalent to $D(s)$, then $\delta_k(\bar{D}) = \delta_k(D)$ and $\zeta_k(\bar{D}) = \zeta_k(D)$ hold.* □

In the Kronecker canonical form of $D(s)$, we have $c = r - \max\{k \mid \zeta_k(D) = 0\}$ and $d = r - \max\{k \mid \delta_k(D) = k\}$. Moreover, we have $\rho_i = \zeta_{r-i+1}(D) - \zeta_{r-i}(D)$ for $i = 1, \ldots, c$ and $\mu_i = \delta_{r-i}(D) - \delta_{r-i+1}(D) + 1$ for $i = 1, \ldots, d$. These equalities imply

$$\sum_{i=1}^{d} \mu_i = r - \delta_r(D), \qquad \sum_{i=1}^{c} \rho_i = \zeta_r(D). \tag{1}$$

Since the sum of the structural indices is equal to the rank of $D(s)$, we have

$$\nu + \sum_{i=1}^{p} \varepsilon_i + \sum_{i=1}^{q} \eta_i = \delta_r(D) - \zeta_r(D). \tag{2}$$

For an $m \times n$ matrix pencil $D(s) = sA + B$, we construct a $(k+1)m \times kn$ matrix $\Psi_k(D)$ and a $km \times (k+1)n$ matrix $\Phi_k(D)$ defined by

$$\Psi_k(D) = \begin{pmatrix} A & O & \cdots & O \\ B & A & \ddots & \vdots \\ O & B & \ddots & O \\ \vdots & \ddots & \ddots & A \\ O & \cdots & O & B \end{pmatrix}, \qquad \Phi_k(D) = \begin{pmatrix} A & B & O & \cdots & O \\ O & A & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & O \\ O & \cdots & O & A & B \end{pmatrix}.$$

We denote $\psi_k(D) = \operatorname{rank} \Psi_k(D)$ and $\varphi_k(D) = \operatorname{rank} \Phi_k(D)$. We also construct a pair of $km \times kn$ matrices $\Theta_k(D)$ and $\Omega_k(D)$ defined by

$$\Theta_k(D) = \begin{pmatrix} A & B & O & \cdots & O \\ O & A & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & O \\ \vdots & & \ddots & A & B \\ O & \cdots & \cdots & O & A \end{pmatrix}, \qquad \Omega_k(D) = \begin{pmatrix} B & O & \cdots & \cdots & O \\ A & B & \ddots & & \vdots \\ O & A & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & B & O \\ O & \cdots & O & A & B \end{pmatrix}.$$

We denote $\theta_k(D) = \operatorname{rank} \Theta_k(D)$ and $\omega_k(D) = \operatorname{rank} \Omega_k(D)$. Then it is easy to see that the ranks of these expanded matrices are expressed by the structural indices as follows.

Theorem 2. *Let $(\nu, \rho_1, \cdots, \rho_c, \mu_1, \ldots, \mu_d, \varepsilon_1, \ldots, \varepsilon_p, \eta_1, \ldots, \eta_q)$ be the structural indices of a matrix pencil $D(s)$. Then we have*

$$\psi_k(D) = rk + \sum_{i=1}^{p} \min\{k, \varepsilon_i\}, \qquad \varphi_k(D) = rk + \sum_{i=1}^{q} \min\{k, \eta_i\},$$

$$\theta_k(D) = rk - \sum_{i=1}^{d} \min\{k, \mu_i\}, \qquad \omega_k(D) = rk - \sum_{i=1}^{c} \min\{k, \rho_i\},$$

where r is the rank of $D(s)$. In particular, $\psi_1(D) = r + p$ and $\varphi_1(D) = r + q$. □

Corollary 1. *If $D(s)$ is of column-full rank, so is $\Psi_k(D)$ for each k. If $D(s)$ is of row-full rank, so is $\Phi_k(D)$ for each k.*

Proof. If $D(s)$ is of column-full rank, the Kronecker canonical form has no minimal column indices. Hence Theorem 2 implies $\psi_k(D) = rk$. Similarly, if $D(s)$ is of row-full rank, the Kronecker canonical form has no minimal row indices, which together with Theorem 2 implies $\varphi_k(D) = rk$. □

Theorem 2 together with (1) implies the following corollary.

Corollary 2. *Let $D(s)$ be a matrix pencil of rank r. For $k \geq r$, we have $\theta_k(D) = r(k-1) + \delta_r(D)$ and $\omega_k(D) = rk - \zeta_r(D)$.*

Proof. For $k \geq r$, we have $\theta_k(D) = rk - \sum_{i=1}^{d} \mu_i$ and $\omega_k(D) = rk - \sum_{i=1}^{c} \rho_i$ by Theorem 2. Then it follows from (1) that $\theta_k(D) = r(k-1) + \delta_r(D)$ and $\omega_k(D) = rk - \zeta_k(D)$ hold for $k \geq r$. □

3 Generic Matrix Pencils

Given a matrix A with the row set R and the column set C, we construct a bipartite graph $G(A) = (R, C; E)$ with the vertex sets R and C and the edge set E that consists of nonzero entries of A. A subset $M \subseteq E$ is called a matching if no two edges in M share an end-vertex. The term-rank of A, denoted by t-rank A, is the maximum size of a matching in $G(A)$. The term-rank provides an upper bound on the rank of A. Under the genericity assumption that the set of nonzero entries are algebraically independent, this upper bound is tight. That is, rank A = t-rank A holds for a generic matrix. A set function τ defined by $\tau(X) =$ t-rank $A[X, C]$ for $X \subseteq R$ is submodular, i.e,

$$\tau(X) + \tau(Z) \geq \tau(X \cup Z) + \tau(X \cap Z)$$

holds for any $X, Z \subseteq R$. This submodularity will be used later in Section 7.

A matrix pencil $D(s) = sA + B$ is called a generic matrix pencil if the nonzero entries in A and B are indeterminates (independent parameters). To be more precise, suppose $D(s)$ is a matrix pencil over a field $\mathbf{F}$. That is, A and B are matrices over the field $\mathbf{F}$. Then $D(s)$ is a generic matrix pencil if the set $\mathcal{T}$ of nonzero entries in A and B is algebraically independent over the prime field $\mathbf{K}$ of $\mathbf{F}$. A typical setting in practice is $\mathbf{F} = \mathbf{R}$ and $\mathbf{K} = \mathbf{Q}$.

For a matrix pencil $D(s) = sA + B$ with the row set R and the column set C, let E and F be the sets of nonzero entries in A and in B, respectively. Then we construct a bipartite graph $G(D) = (R, C; E \cup F)$ with the vertex sets R and C and the edge set $E \cup F$. Note that $G(D)$ allows parallel edges. Each edge e has weight $w(e)$ defined by $w(e) = 1$ for $e \in E$ and $w(e) = 0$ for $e \in F$. A subset M of $E \cup F$ is called a matching if no two edges in M share an end-vertex. The maximum size of a matching in $G(D)$ is the term-rank, denoted by t-rank $D(s)$. The weight $w(M) = \sum_{e \in M} w(e)$ of a matching M is equal to the number of edges in $M \cap E$. We denote by $\widehat{\delta}_k(D)$ the maximum weight of a matching of size k. We also denote by $\widehat{\zeta}_k(D)$ the minimum weight of a matching of size k.

The following two lemmas demonstrate that some fundamental quantities of a generic matrix pencil coincide with their combinatorial counterparts. It should be emphasized here that these combinatorial counterparts are easy to compute with efficient combinatorial algorithms for bipartite matchings.

Lemma 2. *For a generic matrix pencil $D(s)$, we have* rank $D(s) =$ t-rank $D(s)$. □

Lemma 3. *For a generic matrix pencil $D(s)$, we have $\delta_k(D) = \widehat{\delta}_k(D)$ and $\zeta_k(D) = \widehat{\zeta}_k(D)$.* □

4 Dulmage-Mendelsohn Decomposition

In this section, we recapitulate the Dulmage-Mendelsohn decomposition of bipartite graphs following the exposition in [15–§2.2.3].

With a matrix pencil $D(s)$, we associate a submodular function f defined as follows. Let $\Gamma(Y) \subseteq R$ denote the set of vertices adjacent to $Y \subseteq C$ in $G(D)$. Then the function f defined by

$$f(Y) = |\Gamma(Y)| - |Y| \qquad (Y \subseteq C)$$

is submodular, i.e.,

$$f(Y) + f(Z) \geq f(Y \cup Z) + f(Y \cap Z)$$

holds for any $Y, Z \subseteq C$.

The term-rank of the matrix pencil $D(s)$ is characterized by the minimum value of this submodular function f, i.e.,

$$\text{t-rank}\, D(s) = \min\{f(Y) \mid Y \subseteq C\} + |C|,$$

which follows from the Hall-Ore theorem for bipartite graphs. The set of minimizers of a submodular function forms a distributive lattice.

Let Y_0 denote the unique minimal minimizer of f and Y_∞ denote the unique maximal minimizer of f. We put

$$\begin{aligned} C_0 &= Y_0, & R_0 &= \Gamma(Y_0), \\ C_* &= Y_\infty \setminus Y_0, & R_* &= \Gamma(Y_\infty) \setminus \Gamma(Y_0), \\ C_\infty &= C \setminus Y_\infty, & R_\infty &= R \setminus \Gamma(Y_\infty). \end{aligned}$$

Then $D(s)$ is in a block-triangular form with respect to the partitions $(R_0; R_*; R_\infty)$ and $(C_0; C_*; C_\infty)$. That is, $D(s)[R_*, C_0] = O$, $D(s)[R_*, C_0] = O$, $D(s)[R_\infty, C_*] = O$.

Furhtermore, if $D(s)$ is a generic matrix, we have

$$\begin{aligned} \text{rank}\, D(s)[R_0, C_0] &= |R_0|, \\ \text{rank}\, D(s)[R_*, C_*] &= |R_*| = |C_*|, \\ \text{rank}\, D(s)[R_\infty, C_\infty] &= |C_\infty|. \end{aligned}$$

We call $D_0(s) = D(s)[R_0, C_0]$ the horizontal tail and denote its rank by $r_0 = |R_0|$. We also call $D_\infty(s) = D(s)[R_\infty, C_\infty]$ the vertical tail and denote its rank by $r_\infty = |C_\infty|$.

This block-triangularization is called the Dulmage-Mendelsohn decomposition (DM-decomposition) of $D(s)$. The DM-decomposition can be computed efficiently with the aid of the bipartite matching algorithms.

5 The Kronecker Canonical Form via DM-Decomposition

In this section, we investigate the Kronecker canonical form of a generic matrix pencil $D(s)$ via the DM-decomposition.

Lemma 4. *For the horizontal tail D_0 of a generic matrix pencil $D(s)$, we have $\psi_k(D) = \psi_k(D_0) + k(r - r_0)$ and $\widehat{\psi}_k(D) = \widehat{\psi}_k(D_0) + k(r - r_0)$.*

Proof. Recall $r = |R_0| + |C \setminus C_0|$ and $r_0 = |R_0|$. Since $D_*(s) = D(s)[R \setminus R_0, C \setminus C_0]$ is of column-full rank, so is $\Psi_k(D_*)$ by Corollary 1, namely, $\psi_k(D_*) = k|C \setminus C_0| = k(r - r_0)$. Since $\psi_k(D_*) \leq \widehat{\psi}_k(D_*) \leq k(r - r_0)$, we also have $\widehat{\psi}_k(D_*) = k(r - r_0)$. Then it follows from $D(s)[R \setminus R_0, C_0] = O$ that $\psi_k(D) = \psi_k(D_0) + \psi_k(D_*) = \psi_k(D_0) + k(r - r_0)$ and $\widehat{\psi}_k(D) = \widehat{\psi}_k(D_0) + \widehat{\psi}_k(D_*) = \widehat{\psi}_k(D_0) + k(r - r_0)$. □

Lemma 4 together with Theorem 2 implies that the minimal column indices of $D(s)$ coincide with those of $D_0(s)$. We now investigate the Kronecker canonical form of the horizontal tail D_0.

Let $g_0(s)$ be the monic determinantal divisor

$$g_0(s) = \gcd\{\det D(s)[R_0, Y] \mid |Y| = r_0, Y \subseteq C_0\},$$

where gcd designates the greatest common divisor whose leading coefficient is equal to one. The following lemma is a special case of a theorem of Murota [12] (see also [15–Theorem 6.3.8]). We describe its proof here for completeness.

Lemma 5. *The monic determinantal divisor $g_0(s)$ is a monomial in s.*

Proof. We first claim that $g_0(s)$ belongs to $\mathbf{K}[s]$. For any column $j \in C_0$, there exists a column subset $Y \subseteq C_0 \setminus \{j\}$ such that $|Y| = r_0$ and $\det D(s)[R_0, Y] \neq 0$. Therefore, for any independent parameter $t \in \mathcal{T}$, the monic determinantal divisor $g_0(s)$ is free from t. Thus $g_0(s)$ is a polynomial over $\mathbf{K}$.

We now suppose that $g_0(s)$ is not a monomial in s. Let $\overline{\mathbf{K}}$ be the algebraic closure of $\mathbf{K}$. Note that $\mathcal{T}$ is algebraically independent over $\overline{\mathbf{K}}$. Then there exists a root $\xi \in \overline{\mathbf{K}} \setminus \{0\}$ that satisfies $g_0(\xi) = 0$. For a regular submatrix $D(s)[R_0, Y]$, we have $\det D(\xi)[R_0, Y] = 0$, which contradicts the algebraic independence of $\mathcal{T}$ over $\overline{\mathbf{K}}$. □

The determinantal divisor is invariant under strict equivalence transformations. Hence $g_0(s)$ is equal to the determinantal divisor of the Kronecker canonical form $\bar{D}_0$ of D_0. Then Lemma 5 implies that $\bar{D}_0$ does not contain a strictly regular block.

Theorem 3. *The sum of the minimal column indices is obtained by*

$$\sum_{i=1}^{p} \varepsilon_i = \widehat{\delta}_{r_0}(D_0) - \widehat{\zeta}_{r_0}(D_0). \tag{3}$$

Proof. Since the Kronecker canonical form $\bar{D}_0$ of D_0 does not contain a strictly regular block or a rectangular block $L_\eta^\top$, it follows from (2) and Lemma 3 that $\sum_{i=1}^{p} \varepsilon_i = \delta_{r_0}(D_0) - \zeta_{r_0}(D_0) = \widehat{\delta}_{r_0}(D_0) - \widehat{\zeta}_{r_0}(D_0)$. □

A similar argument applied to the vertical tail D_∞ leads to the following results. Lemma 6 implies by Theorem 2 that the minimal row indices of $D(s)$ coincide with those of $D_\infty(s)$. Lemma 7 shows that the Kronecker canonical form $\bar{D}_\infty(s)$ of $D_\infty(s)$ does not contain the strictly regular block.

Lemma 6. *For the vertical tail D_∞ of a generic matrix pencil $D(s)$, we have $\varphi_k(D) = \varphi_k(D_\infty) + k(r - r_\infty)$ and $\widehat{\varphi}_k(D) = \widehat{\varphi}_k(D_\infty) + k(r - r_\infty)$.* □

Lemma 7. *The monic determinantal divisor*

$$g_\infty(s) = \gcd\{\det D(s)[X, C_\infty] \mid X \subseteq R_\infty, |X| = r_\infty\}$$

is a monomial in s. □

Theorem 4. *The sum of the minimal row indices is obtained by*

$$\sum_{i=1}^{q} \eta_i = \widehat{\delta}_{r_\infty}(D_\infty) - \widehat{\zeta}_{r_\infty}(D_\infty). \tag{4}$$

□

As an immediate consequence of Theorems 3 and 4, we have the following theorem implied by (2).

Theorem 5. *The size ν of the strictly regular block in the Kronecker canonical form $\bar{D}(s)$ of $D(s)$ is obtained by*

$$\nu = \widehat{\delta}_r(D) - \widehat{\zeta}_r(D) - \widehat{\delta}_{r_0}(D_0) + \widehat{\zeta}_{r_0}(D_0) - \widehat{\delta}_{r_\infty}(D_\infty) + \widehat{\zeta}_{r_\infty}(D_\infty). \tag{5}$$

□

Note that all these right-hand sides of (3), (4), and (5) can be computed efficiently by the DM-decomposition and weighted bipartite matching algorithms. We have thus obtained a useful combinatorial characterization of the sums of the minimal row/column indices as well as the size of the strictly regular block in the Kronecker canonical form.

Among the structural indices of a generic matrix pencil, $\mu_1, \cdots, \mu_d$ and $\rho_1, \cdots, \rho_c$ are known to be efficiently computable by weighted bipartite matching algorithms. The results in this section enables us to compute ν, $\sum_{i=1}^{p} \varepsilon_i$ and $\sum_{i=1}^{q} \eta_i$ as well. It still remains open to determine the values of the minimal row/column indices. The obtained partial results, however, provide sufficient information to discern if the Kronecker canonical form contains vertical/horizontal rectangular blocks.

6 Expanded Matrices for Indices of Nilpotency

We now turn to the ranks of the expanded matrices, which are in a close relation to the structural indices as shown in Theorem 2, for a generic matrix pencil $D(s) = sA+B$. Even though the set of nonzero entries in A and B is algebraically independent, the expanded matrices are not generic matrices. It will be shown, however, that the ranks of the expanded matrices are equal to their term-ranks in most cases. In this section, we deal with the expanded matrices $\Theta_k(D)$ and $\Omega_k(D)$, which are particularly related to $\mu_1, \cdots, \mu_d$ and $\rho_1, \cdots, \rho_c$. The other expanded matrices $\Psi_k(D)$ and $\Phi_k(D)$ will be investigated in Section 7.

In order to examine the ranks of $\Theta_k(D)$ and $\Omega_k(D)$, we consider the bipartite graphs $G(\Theta_k(D))$ and $G(\Omega_k(D))$ associated with the expanded matrices. It will turn out that these bipartite graphs allow maximum matchings with periodic structures. As a consequence, the ranks of these expanded matrices are equal to their term-ranks denoted by $\widehat{\theta}_k(D)$ and $\widehat{\omega}_k(D)$.

We first investigate $\Theta_k(D)$. Let $\bar{R}$ and $\bar{C}$ be the row set and the column set of $\Theta_k(D)$. Then $\bar{R} = R^1 \cup \cdots \cup R^k$ and $\bar{C} = C^1 \cup \cdots \cup C^k$, where R_h and C_h are the copies of the row set R and the column set C of D for $h = 1, \ldots, k$. For vertices $u \in R$ and $v \in C$, we denote by u^h and v^h the corresponding vertices in R^h and C^h. The edge set of the bipartite graph $G(\Theta_k(D))$ consists of $\bar{E} = E^1 \cup \cdots \cup E^k$ and $\bar{F} = F^1 \cup \cdots \cup F^{k-1}$, where E^h and F^h are the copies of E and F. The edges in E^h connect R^h and C^h, whereas the edges in F^h connect R^h and C^{h+1}. In other words, $E^h = \{(u^h, v^h) \mid (u, v) \in E\}$ and $F^h = \{(u^h, v^{h+1}) \mid (u, v) \in F\}$.

For a matching M in $G(D)$, let M° be the set of edges (u^h, v^h) with $(u, v) \in E \cap M$ for $h = 1, \ldots, k$ and (u^h, v^{h+1}) with $(u, v) \in F \cap M$ for $h = 1, \ldots, k-1$. Then M° forms a matching in $G(\Theta_k(D))$. A matching in $G(\Theta_k(D))$ is called a periodic matching if it can be represented as M° with a certain matching M in $G(D)$.

We now introduce the weight w_k on the edge set of $G(D)$ by $w_k(e) = k$ for $e \in E$ and $w_k(e) = k-1$ for $e \in F$. For a matching M in $G(D)$, we consider the weight of M by $w_k(M) = \sum_{e \in M} w_k(e)$.

Lemma 8. *Let M be a matching that maximizes $w_k(M)$ in $G(D)$. Then the corresponding periodic matching M° is a maximum matching in $G(\Theta_k(D))$.*

Proof. Suppose to the contrary that M° is not a maximum matching in $G(\Theta_k(D))$. Then there exists an augmenting path with respect to M° in $G(\Theta_k(D))$. Let P° be such an augmenting path with minimum number of arcs. The corresponding set of edges in $G(D)$ forms an augmenting path P with respect to M in $G(D)$. Then we analyze the weight of a matching $M' = M \triangle P$, which is the symmetric difference of M and P. Suppose that the end-vertices of P° are $u^h \in R^h$ and $v^l \in C^l$. If $h < l$, then the weight of M' satisfies $w_k(M') - w_k(M) = (l-h)(k-1) - k(l-h-1) = k - l + h$. On the other hand, if $h \geq l$, then we have $w_k(M') - w_k(M) = (h-l+1)k - (h-l)(k-1) = k + h - l$. Thus in either case, we have $w_k(M') = w_k(M) + k + h - k > w_k(M)$, which contradicts the maximality of $w_k(M)$. □

Let M° be a maximum periodic matching in $G(\Theta_k(D))$ corresponding to the maximum-weight matching M in $G(D)$. We denote by ∂M° the set of end-vertices of the edges in M°. Consider the submatrix $\Theta_k(D)[X, Y]$ determined by $X = \bar{R} \cap \partial M$ and $Y = \bar{C} \cap \partial M$. Then the expansion of $\det \Theta_k(D)[X, Y]$ contains a nonzero term

$$\prod_{(u,v)\in M\cap E} {A_{uv}}^k \prod_{(u,v)\in M\cap F} {B_{uv}}^{k-1},$$

where A_{uv} and B_{uv} denote the (u, v)-components of A and B. Each A_{uv} appears exactly k times in $\Theta_k(D)$ and B_{uv} appears exactly $k-1$ times in $\Theta_k(D)$. Hence no other matching cancels this term in the expansion. Thus $\Theta_k(D)[X, Y]$ is a nonsingular submatrix of size $|M^\circ|$, which implies $\theta_k(D) = \widehat{\theta}_k(D)$ by Lemma 8.

A similar argument on $G(\Omega_k(D))$ leads to $\omega_k(D) = \widehat{\omega}_k(D)$. Thus we obtain the following theorem.

Theorem 6. *For a generic matrix pencil $D(s)$, we have $\theta_k(D) = \widehat{\theta}_k(D)$ and $\omega_k(D) = \widehat{\omega}_k(D)$.*

7 Expanded Matrices for Column/Row Indices

This section is devoted to a combinatorial analysis of the ranks of the expanded matrices $\Psi_k(D)$ and $\Phi_k(D)$ for a generic matrix pencil $D(s)$. Let $\widehat{\psi}_k(D)$ and $\widehat{\varphi}_k(D)$ denote the term-ranks of these matrices. Since these expanded matrices admit the same indeterminates to appear in different places, it is not immediately clear that the ranks are equal to the term-ranks. In fact, there is an example of such a expanded matrix whose rank is less than its term-rank. Our analysis, however, makes it possible to assert that they are equal for sufficiently large k as well as $k = 1, 2$.

Theorem 7. *For a generic matrix pencil $D(s)$ of rank r, let r_0 be the rank of the horizontal tail D_0. If $k \geq r_0$, we have*

$$\psi_k(D) = \widehat{\psi}_k(D) = kr + \widehat{\delta}_{r_0}(D_0) - \widehat{\zeta}_{r_0}(D_0).$$

Proof. Due to the submodularity of the term-rank, we have

$$\widehat{\psi}_k(D_0) + \widehat{\varphi}_{k-1}(D_0) \leq \widehat{\theta}_k(D_0) + \widehat{\omega}_k(D_0).$$

Since D_0 is of row-full rank, Corollary 1 implies $\widehat{\varphi}_{k-1}(D_0) = (k-1)r_0$. It follows from Corollary 2 and Theorem 6 that $\widehat{\theta}_k(D_0) = \theta_k(D_0) = r_0(k-1) - \delta_{r_0}(D_0)$ and $\widehat{\omega}_k(D_0) = \omega_k(D_0) = r_0 k - \zeta_{r_0}(D_0)$ hold for $k \geq r_0$ Thus, we obtain

$$\widehat{\psi}_k(D_0) \leq kr_0 + \delta_{r_0}(D_0) - \zeta_{r_0}(D_0).$$

Then by Lemma 4 we have

$$\widehat{\psi}_k(D) \leq kr + \widehat{\delta}_{r_0}(D_0) - \widehat{\zeta}_{r_0}(D_0).$$

On the other hand, Theorems 2 and 3 imply $\psi_k(D) = rk + \widehat{\delta}_{r_0}(D_0) - \zeta_{r_0}(D_0)$ for $k \geq r_0$. Since $\psi_k(D_0) \leq \widehat{\psi}_k(D_0)$, we have $\psi_k(D) = \widehat{\psi}_k(D) = kr + \widehat{\delta}_{r_0}(D_0) - \widehat{\zeta}_{r_0}(D_0)$ for $k \geq r_0$. □

A similar argument applied to the vertical tail D_∞ leads to the following theorem.

Theorem 8. *For a generic matrix pencil $D(s)$ of rank r, let r_∞ be the rank of the vertical tail D_∞. If $k \geq r_\infty$, we have*

$$\varphi_k(D) = \widehat{\varphi}_k(D) = kr + \widehat{\delta}_{r_\infty}(D_\infty) - \widehat{\zeta}_{r_\infty}(D_\infty). \qquad \square$$

For $k = 1, 2$, the ranks of the expanded matrices coincide with their term-ranks. This is immediate for $k = 1$ as Ψ_1 and Φ_1 are generic matrices. The following theorem deals with the case of $k = 2$.

Theorem 9. *For a generic matrix pencil $D(s)$, we have $\psi_2(D) = \widehat{\psi}_2(D)$ and $\varphi_2(D) = \widehat{\varphi}_2(D)$.*

Proof. Let M^* be a maximum matching in $G(\Psi_2(D))$. The row set $\bar{R}$ and the column set $\bar{C}$ are given by $\bar{R} = R_1 \cup R_2 \cup R_3$ and $\bar{C} = C_1 \cup C_2$, where R_h and C_h are the copies of R and C. For $u \in R$ and $v \in C$, we denote their copies by $u_h \in R_h$ and $v_h \in C_h$. We also denote $X = \bar{R} \cap \partial M^*$ and $Y = \bar{C} \cap \partial M^*$. Then it suffices to show that $W = \Psi_2(D)[X, Y]$ is nonsingular.

Let M_1^* and M_2^* be the sets of edges in M^* incident to C_1 and C_2, respectively. We denote $X_1 = X \cap \partial M_1^*$, $X_2 = X \cap \partial M_2^*$, $Y_1 = C_1 \cap \partial M^*$, and $Y_2 = C_2 \cap \partial M^*$.

Let $\mathcal{P}$ denote the family of perfect matchings in $G(W)$. For each matching $M \in \mathcal{P}$, we denote $\pi(M) = \prod_{(u,v)\in M} W_{uv}$, where W_{uv} is the (u, v)-component of W. Recall that

$$\det W = \sum_{M \in \mathcal{P}} \sigma_M \pi(M),$$

where σ_M takes 1 or -1. We also denote by $\mathcal{P}^\bullet$ the family of perfect matchings $M = M_1 \cup M_2$ such that $\partial M_1 = \partial M_1^*$ and $\partial M_2 = \partial M_2^*$.

We now claim that $\pi(M^\bullet) \neq \pi(M')$ for any pair of $M^\bullet \in \mathcal{P}^\bullet$ and $M' \in \mathcal{P} \backslash \mathcal{P}^\bullet$. Suppose to the contrary that $\pi(M^\bullet) = \pi(M')$. The matching M' contains an edge (u_2, v_2) with $u_2 \in X_1$. Then we have $(u_1, v_1) \in M^\bullet \backslash M'$, and hence $(u_1, z_1) \in M'$ for some $z \in C \setminus \{v\}$, which implies $(u_2, z_2) \in M^\bullet$. This is a contradiction to $u_2 \in X_1$.

Since both $W[X_1, Y_1]$ and $W[X_2, Y_2]$ are generic matrices, we have

$$\sum_{M \in \mathcal{P}^\bullet} \sigma_M \pi(M) = \det W[X_1, Y_1] \cdot \det W[X_2, Y_2] \neq 0.$$

Then the claim above implies $\det W \neq 0$, which means W is nonsingular. $\square$

For general k, however, the ranks of the expanded matrices may differ from their term-ranks. For instance, consider a generic matrix pencil

$$D(s) = sA + B = \begin{pmatrix} \beta_1 & s\alpha_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \beta_2 & s\alpha_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & s\alpha_3 & \beta_3 & 0 & 0 & 0 \\ 0 & 0 & \beta_4 & 0 & s\alpha_4 & \beta_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & s\alpha_5 + \beta_6 & s\alpha_6 & \beta_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & \beta_8 & s\alpha_7 \end{pmatrix}.$$

The Kronecker canonical form of $D(s)$ is $\bar{D}(s) = \text{block-diag}(L_4, L_2)$, which implies

$$\psi_k(D) = \begin{cases} 8k & (k \leq 2) \\ 7k+2 & (2 \leq k \leq 4) \\ 6k+6 & (k \geq 4), \end{cases}$$

whereas

$$\widehat{\psi}_k(D) = \begin{cases} 8k & (k \leq 3) \\ 6k+6 & (k \geq 3). \end{cases}$$

Thus $\psi_k(D) = \widehat{\psi}_k(D)$ holds for $k \neq 3$. For $k = 3$, however, we have $\psi_3(D) = 23$ and $\widehat{\psi}_3(D) = 24$.

8 Generic Matrix Powers

As a byproduct of our combinatorial analysis in Section 6, we give a simple alternative proof for a theorem of Poljak [17] on the ranks of powers of generic square matrices.

Let A be an $n \times n$ generic matrix. We associate a directed graph $\boldsymbol{G}(A) = (R, \boldsymbol{E})$ with vertex set R identical with the row/column set of A. The arc set $\boldsymbol{E}$ is the set of nonzero entries of A, namely $\boldsymbol{E} = \{(u, v) \mid A_{uv} \neq 0\}$. A k-walk in $\boldsymbol{G}(A)$ is an alternating sequence $(v_0, e_1, v_1, \cdots, e_k, v_k)$ of vertices $v_h \in R$ and $e_h \in \boldsymbol{E}$ such that $e_h = (v_{h-1}, v_h)$ for $h = 1, \ldots, k$. A pair of k-walks $(v_0, e_1, v_1, \cdots, e_k, v_k)$ and $(v'_0, e'_1, v'_1, \cdots, e'_k, v'_k)$ is called independent if $v_h \neq v'_h$ holds for $h = 0, 1, \ldots, k$. The following theorem characterizes the rank of A^k in terms of independent k-walks.

Theorem 10 (Poljak [17]). *For a generic square matrix A, the rank of A^k is equal to the maximum number of mutually independent k-walks in $\boldsymbol{G}(A)$.*

Consider a regular matrix pencil $D(s) = sA + I$, where I denotes the unit matrix. Then a k-walk naturally corresponds to a path P in $G(\Theta_k(D))$ from C^1 to R^k. To be more specific, the path P is given by

$$P = \{({v_{h-1}}^h, \bar{v_h}^h) \mid h = 1, \ldots, k\} \cup \{({v_h}^h, \bar{v_h}^h) \mid h = 1, \ldots, k\},$$

where $\bar{v_h}$ denotes the column that is identical to the row $v_h \in R$. Then a pair of independent k-walks correspond to a pair of vertex-disjoint paths in $G(\Theta_k(D))$. Let $\bar{P} = P_1 \cup \cdots \cup P_\ell$ denotes the edge set of ℓ such vertex-disjoint paths that come from ℓ independent k-walks. Then the symmetric difference $\bar{P} \triangle \bar{F}$ forms a matching of size $\ell + n(k-1)$. Conversely, any periodic matching M° can be obtained in this way from a set of independent k walks. Therefore, Lemma 8 implies that the maximum number of independent k-walks is equal to $\widehat{\theta}_k(D) - (k-1)n$.

On the other hand, we have

$$\operatorname{rank} A^k = \theta_k(D) - (k-1)n.$$

Therefore, in order to prove Theorem 10, it suffices to show that $\theta_k(D) = \widehat{\theta}_k(D)$. Since $D(s)$ is not a generic matrix, we can not directly apply Theorem 6. However, we can use essentially the same argument.

Let M° be a maximum periodic matching in $G(\Theta_k(D))$ that corresponds to a matching M in $G(D)$. Consider the submatrix $\Theta_k(D)[X, Y]$ with $X = \bar{R} \cap \partial M$ and $Y = \bar{C} \cap \partial M$. Then the expansion of $\det \Theta_k(D)[X, Y]$ contains a nonzero term

$$\prod_{(u,v)\in M\cap E} {A_{uv}}^k,$$

where A_{uv} denotes the (u, v)-component of A. Since each A_{uv} appears exactly k times in $\Theta_k(D)$, no other matching cancels this term in the expansion. Thus $\Theta_k(D)[X, Y]$ is a nonsingular submatrix of size $|M^\circ|$, which implies $\theta_k(D) = \widehat{\theta}_k(D)$ by Lemma 8.

9 Conclusion

This paper has investigated the Kronecker canonical form of generic matrix pencils. Even if the genericity assumption is not valid, we can efficiently compute the combinatorial estimates of the sums of the minimal row/column indices. These estimates are correct in most cases unless there is an unlucky numerical cancellation. Hence we can use them for checking if the result of numerical computation is consistent with the combinatorial information.

Another way to use the combinatorial estimates is to design a numerical algorithm that exploits the combinatorial information. If one had an easier way to check the correctness of the estimates, it would lead to a new algorithm particularly efficient for sparse matrices. In fact, such algorithms of combinatorial relaxation type have been developed for the maximum degree of subdeterminants [9, 10, 13, 15]. It would be interesting to devise the same type of algorithms for minimal row/column indices.

References

1. T. Beelen and P. Van Dooren, An improved algorithm for the computation of Kronecker's canonical form, *Linear Algebra Appl.*, 105 (1988), pp. 9–65.
2. J. Demmel and B. Kågström, Accurate solutions of ill-posed problems in control theory, *SIAM J. Matrix Anal. Appl.*, 9 (1988), pp. 126–145.
3. J. Demmel and B. Kågström, The generalized Schur decomposition of an arbitrary pencil $A - B$: Robust software with error bounds and applications, *ACM Trans. Math. Software*, 19 (1993), pp. 160–201.
4. I. Duff and C. W. Gear, Computing the structural index, *SIAM J. Algebraic Discrete Meth.*, 7 (1986), pp. 594–603.
5. A. Edelman, E. Elmroth, and B. Kågström, A geometric approach to perturbation theory of matrices and matrix pencils. Part I: Versal deformations, *SIAM J. Matrix Anal. Appl.*, 18 (1997), pp.653–692.

6. A. Edelman, E. Elmroth, and B. Kågström, A geometric approach to perturbation theory of matrices and matrix pencils. Part II: A stratification-enhanced staircase algorithm, *SIAM J. Matrix Anal. Appl.*, 20 (1999), pp.667–699.
7. C. W. Gear, Differential-algebraic equation index transformations, *SIAM J. Sci. Stat. Comput.*, 9 (1988), pp. 39–47.
8. C. W. Gear, Differential algebraic equations, indices, and integral algebraic equations, *SIAM J. Numer. Anal.*, 27 (1990), pp. 1527–1534.
9. S. Iwata, Computing the maximum degree of minors in matrix pencils via combinatorial relaxation, *Algorithmica*, 36 (2003), pp. 331–341.
10. S. Iwata, K. Murota and I. Sakuta, Primal-dual combinatorial relaxation algorithms for the maximum degree of subdeterminants, *SIAM J. Sci. Comput.*, 17 (1996), pp. 993–1012.
11. B. Kågström, RGSVD — An algorithm for computing the Kronecker canonical form and reducing subspaces of singular matrix pencils $A - B$, *SIAM J. Sci. Stat. Comput.*, 7 (1986), pp. 185–211.
12. K. Murota, *On the Smith normal form of structured polynomial matrices*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 747–765.
13. K. Murota, Combinatorial relaxation algorithm for the maximum degree of subdeterminants — Computing Smith-McMillan form at infinity and structural indices in Kronecker form, *Appl. Algebra Engin. Comm. Comput.*, 6 (1995), pp. 251–273.
14. K. Murota, On the degree of mixed polynomial matrices, *SIAM J. Matrix Anal. Appl.*, 20 (1999), 196–227.
15. K. Murota, *Matrices and Matroids for Systems Analysis*, Springer-Verlag, 2000.
16. C. C. Pantelides, The consistent initialization of differential-algebraic systems, *SIAM J. Sci. Stat. Comput.*, 9 (1988), pp. 213–231.
17. S. Poljak, Maximum rank of powers of a matrix of a given pattern, *Proc. Amer. Math. Soc.*, 106 (1989), pp.1137–1144.
18. H. H. Rosenbrock, *State-Space and Multivariable Theory*, John Wiley, 1970.
19. J. W. van der Woude, The generic canonical form of a regular structured matrix pencil, Linear Algebra Appl., 353 (2002), pp. 267–288.
20. P. Van Dooren, The computation of Kronecker's canonical form of a singular pencil, *Linear Algebra Appl.*, 27 (1979), pp. 103–140.
21. J. H. Wilkinson, Kronecker's canonical form and the QZ algorithm, *Linear Algebra Appl.*, 28 (1979), pp. 285–303.

Power Optimization for Connectivity Problems

Mohammad T. Hajiaghayi[1], Guy Kortsarz[2], Vahab S. Mirrokni[1], and Zeev Nutov[3]

[1] Computer Science and Artificial Intelligence Laboratory, MIT
{mirrokni, hajiagha}@csail.mit.edu
[2] Department of Computer Science, Rutgers University-Camden
guyk@crab.rutgers.edu
[3] Computer Science Division, The Open University of Israel
nutov@openu.ac.il

Abstract. Given a graph with costs on the edges, the power of a node is the maximum cost of an edge leaving it, and the power of the graph is the sum of the powers of the nodes of this graph. Motivated by applications in wireless multi-hop networks, we consider four fundamental problems under the power minimization criteria: the *Min-Power b-Edge-Cover* problem (MPb-EC) where the goal is to find a min-power subgraph so that the degree of every node is at least some given integer b( ), the *Min-Power k-node Connected Spanning Subgraph problem* (MPk-CSS), *Min-Power k-edge Connected Spanning Subgraph problem* (MPk-ECSS), and finally the *Min-Power k-Edge-Disjoint Paths* problem in directed graphs (MPk-EDP). We give an $O(\log^4$)-approximation algorithm for MPb-EC. This gives an $O(\log^4$)-approximation algorithm for MPk-CSS for most values of k, improving the best previously known $O(k)$-approximation guarantee. In contrast, we obtain an $O(\sqrt{\ })$ approximation algorithm for MPk-ECSS, and for its variant in directed graphs (i.e., MPk-EDP), we establish the following inapproximability threshold: MPk-EDP cannot be approximated within $O(2^{\log^{1-\varepsilon} n})$ for any fixed > 0, unless NP-hard problems can be solved in quasi-polynomial time.

1 Introduction

Wireless multihop networks are an important subject of study due to their extensive applications (see e.g., [8,24]). A large research effort focused on performing network tasks while minimizing the power consumption of the radio transmitters of the network. In ad-hoc networks, a range assignment to radio transmitters means to assign a set of powers to mobile devices. We consider finding a range assignment for the nodes of a network such that the resulting communication network satisfies some prescribed properties, and such that the total power is minimized. Specifically, we consider "min-power" variants of three extensively studied "min-cost" problems: the b-Edge Cover problem and the k-Connected Spanning Subgraph Problem in undirected networks, and the k-Edge-Disjoint Paths problem in directed networks.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 349–361, 2005.

In wired networks, generally we want to find a subgraph with the minimum cost instead of the minimum power. This is the main difference between the optimization problems for wired versus wireless networks. The power model for undirected graphs corresponds to the static symmetric multi-hop ad-hoc wireless networks with omnidirectional transmitters. This model is justified and used in several other papers [3, 4, 14].

An important network task is assuring high fault-tolerance ([1–4, 11, 18]). The simplest version is when we require the network to be connected. In this case, the min-cost variant is just the min-cost spanning tree problem, while the min-power variant is NP-hard even in the Euclidean plane [9]. There are several localized and distributed heuristics to find the range assignment to keep the network connected [18, 24, 25]. Constant approximation guarantees for the min-power spanning tree problem are given in [4, 14]. For general k, the best previously known approximation ratio for MPk-CSS was $2k = O(k)$ [5, 11, 19].

Min-cost k-connected and k-edge connected spanning subgraph problems were extensively studied [7, 10, 12, 13, 16, 17]. While the min-cost k-edge connected spanning subgraph problem admits a 2-approximation algorithm [12,13], no constant approximation guarantee is known for the min-cost k-connected spanning subgraph problem. The best known approximation ratios for the latter are $O(\ln k \cdot \min\{\sqrt{k}, \frac{n+k}{n-k} \ln k\})$ [17] and $O(\ln k)$ for $n \geq 2k^2$ [7].

The notation and preliminaries used in the paper are as follows. Let $G = (V, E)$ be a graph. For disjoint $X, Y \subseteq V$ let $\delta_G(X, Y) = \delta_E(X, Y)$ be the set of edges from X to Y in E. We will often omit the subscripts G, E if they are clear from the context. For brevity, $\delta_E(X) = \delta_E(X, V - X)$, and $\deg_E(X) = |\delta_E(X)|$ is the *degree of* X. For a function g on a groundset U and $S \subseteq U$ let $g(S) = \sum_{u \in S} g(u)$. Given edge costs $c(e)$, $e \in E$, the *power* $p_G(v) = p_E(v)$ *of a node* v in G is the maximum cost of an edge incident to v in E, that is, $p(v) = \max_{e \in \delta_E(v)} c(e)$. The *power of* G is $p(G) = p_E(V) = \sum_{v \in V} p(v)$. Note that $p(G)$ differs from the ordinary cost $c(G) = \sum_{e \in E} c(e)$ of G even for unit costs. In this case, if G has no isolated nodes then $c(G) = |E|$ and $p(G) = |V|$. For example, if E is a perfect matching on V then $p(G) = 2c(G)$. If G is a clique then $p(G)$ is roughly $c(G)/\sqrt{m/2}$[1]. The following statement whose proof is presented in Section 3 shows that these are the extremal cases also for general edge costs.

Lemma 1. *For any graph* $G = (V, E)$ *holds:* $c(G)/\sqrt{|E|/2} \leq p(G) \leq 2c(G)$. *For a forest* T, $c(T) \leq p(T) \leq 2c(T)$.

Throughout the paper, let $\mathcal{G} = (V, \mathcal{E})$ denote the input graph with nonnegative costs on the edges; n denotes the number of nodes in $\mathcal{G}$, and m the number of edges in $\mathcal{G}$. Let opt denote the optimal solution value of an instance at hand. Given $\mathcal{G}$, our goal is to find a minimum power spanning subgraph G of $\mathcal{G}$ that satisfies some prescribed property. In undirected graphs, we consider the following two variants. Given an integral function b on V, we say that G (or E) is

[1] In this paper, we ignore that some numbers might not be integers, since the adaption to floors and ceilings is immediate.

a *b-edge cover* if $\deg_G(v) \geq b(v)$ for every $v \in V$, where $\deg_G(v) = \deg_E(v)$ is the degree of v in G. In the *Minimum Power b-Edge Cover Problem* (MPb-EC), G is required to be a b-edge cover; the *Minimum Power k-Edge Cover Problem* (MPk-EC) is a particular case when $b(v) = k$ for all $v \in V$. It is easy to see that the greedy algorithm that for every $v \in V$ picks the lightest $b(v)$ edges incident to v is a $(k+1)$-approximation algorithm for MPb-EC, where $k = \max_{v \in V} b(v)$. The following simple example shows that the $(k+1)$-approximation ratio is tight for this greedy algorithm. Take $k+1$ stars with k leaves each, and add edges of a clique on their centers. All edges have unit costs. Set $b(v) = k$ if v is a star center and $b(v) = 0$ otherwise. The greedy algorithm may pick the edges of the stars, thus getting a solution of value $(k+1)^2$. The optimal solution is obtained by picking the clique, and has power $k+1$. This example easily extends to MPk-EC. We prove:

Theorem 1. MPb-EC *is APX-hard. It admits an* $O(\log^4 n)$*-approximation algorithm.*

A graph G is *k-(node) connected* if there are k internally disjoint paths between every pair of its nodes. In the *Minimum Power k-Connected Spanning Subgraph Problem* (MPk-CSS) G is required to be k-connected. The motivation of the "min-power" variant for wireless networks is similar to the one of the "min-cost" variants for wired networks, e.g., for MPk-CSS we require that the network remains connected even in failure of up to $k-1$ terminals. The problem admits an $O(k)$-approximation algorithm [5, 11]. We prove:

Theorem 2. MPk-CSS *is APX-hard. Unless* $k = n - o(n)$, MPk-CSS *admits an* $O(\log^4 n)$*-approximation algorithm. For* $k = n - o(n)$, MPk-CSS *admits an* $O(\sqrt{k})$*-approximation algorithm.*

Theorem 2 is proved by combining Theorem 1 with part (i) of the following theorem, and using the currently best known approximation guarantees [7, 16] for the Min-Cost k-Connected Spanning Subgraph problem.

Theorem 3. *(i) If there exists an* α*-approximation algorithm for the Min-Cost k-Connected Spanning Subgraph problem and a* β*-approximation algorithm for* MPk-EC *then there exists a* $(2\alpha + \beta)$*-approximation algorithm for* MPk-CSS.

(ii) If there exists a ρ*-approximation for* MPk-CSS *then there exists a* $(2\rho + 1)$*-approximation for the Min-Cost k-Connected Spanning Subgraph problem.*

Note that part (ii) of Theorem 3 implies that MPk-CSS is almost as hard to approximate as the Min-Cost k-Connected Spanning Subgraph problem.

We also consider the *Min-Power k-Edge Connected Spanning Subgraph* (MPk-ECSS) problem where G is required to be k-edge connected. This problem admits a $O(k)$-approximation algorithm [11]. We prove:

Theorem 4. MPk-ECSS *is APX-hard and admits an* $O(\sqrt{n})$*-approximation algorithm.*

Power optimization problems were considered in asymmetric networks as well [14]. This setting is mainly motivated for the purpose of broadcasting or multicasting in multihop wireless networks. In this case, the power of a node v is the maximum cost of an edge outgoing from v. We give some evidence that minimum-power connectivity problems in directed graphs are hard by showing a strong inapproximability result for a simple variant: the problem of finding the minimum-power subgraph that contains k edge-disjoint directed (s,t)-paths. We call it the *Min-Power k-Edge-Disjoint Paths* (MPk-EDP) problem, since it is the "min-power variant" of the Min-Cost k-Edge-Disjoint Paths problem. We prove the following strong inapproximability result for MPk-EDP, in contrast to the polynomial solvability of the "min-cost" case.

Theorem 5. MPk-EDP *cannot be approximated within* $O(2^{\log^{1-\varepsilon} n})$ *for any fixed* $\varepsilon > 0$*, unless NP-hard problems can be solved in quasi-polynomial time.*

We also note that, in contrast, the problem of finding Minimum Power k-Vertex-Disjoint Paths (from s to t) in directed graphs can be solved in polynomial time as follows. First, we can assume that we know the power p of s (there are at most n possible values) and thus we know all optimum edges incident to s. Now, we give zero cost to all these edge (whose original costs were at most p), delete all the other edges incident to s, and compute the minimum cost k internally vertex-disjoint paths using the polynomial-time min-cost k-flow algorithm of Orlin [21], and a flow decomposition. As the outdegree of every internal node is one, and the outdegree of t is zero, this is an optimal solution to our minimum power vertex-disjoint case.

Table 1 summarizes our main results.

Table 1. Our approximation ratios and hardness results (is the best approximation ratio for the Min-Cost k-Connected Spanning Subgraph problem)

Problem	Approximation Ratio	Hardness
MPb-EC	$\min(O(\log^4 \quad), k+1)$	APX-hard
MPk-CSS	$\min(O(\log^4 \quad)+2 \quad, k(1+ \quad(1))$	APX-hard, ( )
MPk-ECSS	$O(\sqrt{\ })$	APX-hard
MPk-EDP	–	$\left(2^{\log^{1-\varepsilon} n}\right)$

Theorem 1 is proved in Section 2, Lemma 1 and Theorems 2, 3, and 4 are proved in Section 3, and finally Theorem 5 is proved in Section 4. In the rest of this section we show that already very restricted instances of MPb-EC, MPk-CSS, and MPk-ECSS are APX hard, thus proving the hardness results of Theorem 1, 2, and 4.

Theorem 6. MPk-EC*,* MPk-CSS*, and* MPk-ECSS *are APX-hard even for* $k = 1$*.*

Proof. To prove the theorem, we will use the following well-known formulation of the Set-Cover Problem (SCP); in this formulation, J is the incidence graph

of sets and elements, where A is the family of sets and B is the universe (we denote the edge set by I).
Input: A bipartite graph $J = (A \cup B, I)$ without isolated nodes.
Output: A minimum size subset $T \subseteq A$ such that every node in B has a neighbor in T.

The reduction is as follows. Given an instance $J = (A \cup B, I)$ for the SCP, we construct a graph $G = (V \cup \{r\}, E)$ with edge cost function c by setting $c(e) = 1$ for every $e \in E$, adding a new node r and edges of cost zero from r to every $a \in A$; for MPk-EC we set $b(v) = 1$ for every $v \in V$, and for MPk-CSS and MPk-ECSS we set $k = 1$. It is easy to see that SCP has a solution of size τ if and only if the obtained instance of MPk-EC (MPk-CSS/MPk-ECSS) has a solution of size $|B| + \tau$.

A 4-bounded instance SC-4 of SC is one in which all sets have size at most 4, that is $\deg_J(a) \leq 4$ for every $a \in A$. Any solution to SC-4 has size $\geq |B|/4$. Thus any solution of power $|B| + \tau$ in our MPk-EC (MPk-CSS/MPk-ECSS) instance of power at most $(1 + \varepsilon)(|B| + \tau)$ gives us a solution to SC-4 of size at most $\tau + \tau\varepsilon + |B|\varepsilon \leq (1+5\varepsilon)\tau$. Consequently, a $(1+\varepsilon)$-approximation to MPk-EC gives a $(1 + 5\varepsilon)$-approximation to SC-4. Since SC-4 is APX-hard [22], APX-hardness of MPk-EC (MPk-CSS/MPk-ECSS) follows.

Finally to obtain APX-hardness of MPk-EC (MPk-CSS/MPk-ECSS) for $k > 1$, we add vertices $t_1, \ldots, t_{k-1}$ to graph G constructed above and we add edges of zero cost from them to all previous vertices. The proof again follows from the fact that each vertex corresponding to an element should be adjacent to at least one edge of cost one. □

2 Proof of Theorem 1

In this section, we present the proof of Theorem 1. Throughout this section we assume that $c(e) \in \{1, \ldots, n^4\}$ for every $e \in \mathcal{E}$. In particular, opt $\leq n^6$. Indeed, let c be the least integer so that $\{e \in E : u_e \leq c\}$ is a b-edge cover. Edges of cost $\geq cn^2$ do not belong to any optimal solution, and thus deleted from the graph. Edges of cost $\leq c/n^2$ in fact get zero costs, as adding all of them to the solution affects only the constant in the approximation ratio (we also update $b(v)$'s, $v \in V$, accordingly). This gives an instance with $c_{\max}/c_{\min} \leq n^4$, where $c_{\max}$ and $c_{\min}$ denote the maximum and the minimum nonzero cost of an edge in $\mathcal{E}$, respectively. Further, for every $e \in \mathcal{E}$ set $c(e) \leftarrow \lceil c(e)/c_{\min} \rceil$. It is easy to see that the loss incurred in the approximation ratio is only a constant, which is negligible in our context.

Let $b(V) = \sum_{v \in V} b(v)$. For an edge set F and $v \in V$, let

$$b_F(v) = \max\{b(v) - \deg_F(v), 0\}$$

be the *residual deficiency* of v w.r.t. F (so $b(v) = b_\emptyset(v)$). Also, $b_F(V) = \sum_{v \in V} b_F(v)$. Our algorithm runs with a parameter τ that should be set to $\tau =$ opt to achieve the claimed approximation ratio. Specifically, we will prove:

Lemma 2. *There exists a polynomial time algorithm that given an instance of* MPb-EC *and an integer* τ*, either returns an edge set* $E' \subseteq \mathcal{E}$ *such that*

$$p_{E'}(V) = \tau \cdot O(\lg^4 n) \tag{1}$$

$$b_{E'}(V) \leq \lg^3 n \tag{2}$$

or establishes that $\tau < opt$*.*

Note that if $\tau <$ opt, the algorithm may return an edge set E' that satisfies (1) and (2). Let us now show that Lemma 2 implies Theorem 1. Since opt is not known, we apply binary search to find the minimum integer τ so that an edge set E' satisfying (1) and (2) is returned; then $p_{E'}(V) = \text{opt} \cdot O(\lg^4 n)$ (note that binary search for appropriate τ requires $O(\lg n^6) = O(\lg n)$ iterations). Then we apply the greedy algorithm on $\mathcal{G} - E'$ to compute a $b_{E'}$-edge cover E'' of power $\leq$ opt $\cdot$ $(\lg^3 n + 1)$. Then $E = E' \cup E''$ is a feasible solution, and $p_{E' \cup E''}(V) \leq p_{E'}(V) + p_{E''}(V) = \text{opt} \cdot O(\lg^4 n)$. Thus Lemma 2 implies Theorem 1.

The proof of Lemma 2 follows. Let $D(F) = \{v \in V : b_F(v) > 0\}$ be the set of *deficient* nodes w.r.t. F, and $D = D(\emptyset) = \{v \in V : b(v) > 0\}$. Let $\mu = \min\{b(v) : v \in D\}$.

Lemma 3. *There exists a polynomial time algorithm that given an instance of* MPb-EC *with* $\max\{b(v) : v \in D\} \leq r\mu$ *and integers* W*,* T*, and* τ*, returns an edge set* F *such that*

$$p(F) \leq 2\,(W|D| + b(V) \lg W/T)\,, \tag{3}$$

and if $\tau \geq opt$ *then*

$$b_F(V) \leq \tau\,(T \lg W + \mu r/(2W))\,. \tag{4}$$

Proof. Let $E_0 = \{e \in E : 1 \leq c(e) \leq 2\}$ and $E_i = \{e \in E : 2^i + 1 \leq c(e) \leq 2^{i+1}\}$ for $i = 1, \ldots, \lg W$. Consider the following algorithm that starts with $F = \emptyset$:
For $i = 0$ to $\lg W$ do:
While there is $v \in V$ with $|\delta_{E_i}(v, D(F))| \geq 2^i T$ do $F \leftarrow F + \delta_{E_i}(v, D(F))$
End For
It is easy to see that the algorithm is polynomial. Let F be the edge set computed by the algorithm. Let $p' = \sum_{v \in D} p_F(v)$ and $p'' = \sum_{v \in V - D} p_F(v) = p_F(V) - p'$. The following two claims show that (3) holds.

Claim: $p' \leq 2W|D|$.

Proof: $p_F(v) \leq 2^{i+1} \leq 2W$ for every $v \in D$. Thus $p' \leq 2W|D|$. □

Claim: $p'' \leq 2b(V) \lg W/T$.

Proof: If at iteration i we added to F edges incident to v, then the deficiency of v drops by at least $2^i T$. Thus the total number of nodes in $V - D$ incident to edges added at iteration i is at most $b(V)/(2^i T)$. Since every added edge

has cost at most 2^{i+1} the total increase in the power at iteration i is at most $2^{i+1}b(V)/(2^iT) = 2b(V)/T$. The claim follows. □

Assume that $\tau \geq opt$. Let O be a feasible solution with $p(O) \leq \tau$. Let $A = \{e \in O - F : c(e) \leq 2W\}$, $B = (O - F) - A$. The following two claims show that $O - F$ decreases the deficiency of $D(F)$ by at most $\tau\,(T \lg W + r\mu/2W)$. This implies (4), since $b_F(V) = b_F(D(F))$.

Claim: A decreases the deficiency of $D(F)$ by at most $\tau T \lg W$.

Proof: Fix some $i \leq \lg W$. Let $A_i = E_i \cap A$. Since $p_{A_i}(V) \leq \tau$, the edges in A_i are incident to at most $\tau/2^i$ nodes. Note that $|\delta_{A_i}(v, D(F))| \leq T2^i$ for every $v \in V$. Thus each A_i reduces the deficiency of $D(F)$ by at most τT. The claim follows. □

Claim: B decreases the deficiency of $D(F)$ by at most $\mu r\tau/(2W)$.

Proof: The number of nodes in $D(F)$ adjacent to the edges in B is at most $\tau/(2W)$. The deficiency of each $v \in D(F)$ is at most $r\mu$. The claim follows. □

The proof of Lemma 3 is complete. □

Corollary 1. *There exists a polynomial time algorithm that given an instance of* MPb-EC *with* $\max\{b_v : v \in D\} \leq r\mu$ *and an integer* τ*, returns an edge set* F *such that:* $p(F) = \tau \cdot O(r + \lg^2 n)$ *and if* $\tau \geq opt$ *then* $b_F(V) \leq b(V)/2$.

Proof. For $W = 2\tau\mu r/b(V)$ and $T = b(V)/(4\tau \lg W)$, the algorithm from Lemma 3 computes an edge set F such that (note that $W = 2\tau \cdot (\mu r)/b(V) \leq 2\tau \leq 2n^6$):

$$p(F) \leq 2\left(2\tau r\frac{\mu|D|}{b(V)} + 4\tau \lg^2 W\right) \leq 4\tau\left(r + 2\lg^2 W\right) = \tau \cdot O(r + \lg^2 n).$$

If $\tau \geq opt$ then:

$$b_F(V) \leq \tau\left(\frac{b(V)/4}{\tau} + \frac{\mu r b(V)}{4\tau\mu r}\right) = b(V)\left(\frac{1}{4} + \frac{1}{4}\right) = \frac{b(V)}{2}.$$

□

Proof of Lemma 2: Consider the following algorithm that starts with $E' = \emptyset$:

Algorithm b-Edge-Cover(τ)
While $b(V) \geq \lg^3 n$ do:

- Let $V_0 = \{v \in V : 1 \leq b(v) \leq 2\}$ and $V_j = \{v \in V : 2^j + 1 \leq b(v) \leq 2^{j+1}\}$, $j = 1, \ldots, \lg n$.
- Let q be an index so that $b(V_q) \geq b(V)/\lg n$.
- Compute F as in Corollary 1 with $b'(v) = b(v)$ if $v \in V_q$ and $b'(v) = 0$ otherwise.
- *If* $b_F(V_q) \leq b(V_q)/2$ then: $E' \leftarrow E' \cup F$, $G \leftarrow G - F$, $b \leftarrow b_F$; *Else* declare "$\tau <$ opt" and STOP.

End While

If the algorithm declares "$\tau < \text{opt}$" then this is correct, by Corollary 1. Let us assume therefore that this is not so.

Claim: The algorithm calls to the algorithm from Corollary 1 $O(\lg^2 n)$ times.

Proof: Let B_t be the total residual deficiency before iteration $t+1$ of the while loop, where $B_0 = b(V) \leq n^2$. We have $B_{t+1} \leq B_t(1 - 1/(2\lg n))$, so $B_t \leq B_0(1 - 1/(2\lg n))^t$. Thus after at most

$$\frac{\lg(n^2/\lg^3 n)}{-\lg(1 - 1/(2\lg n))} = O(\lg^2 n)$$

iterations the condition in the while loop is met, and the iterations stop. □

From the last claim and Corollary 1, we obtain that $p_{E'}(V) = \tau \cdot O(\lg^4 n)$ (note that when **Algorithm *b*-Edge-Cover(τ)** calls Corollary 1, we can set $r = 2$ since $b(v)$'s of all $v \in V_q$ are within a factor 2 of each other). □

The proof of Theorem 1 is now complete.

3 Proof of Lemma 1 and Theorems 2–4

We first present the proof of Lemma 1 which is a basis to our results.

Proof of Lemma 1: Except the inequality $c(G)/\sqrt{|E|/2} \leq p(G)$ the statement was proved in [11, 19]. We restate the proof for completeness of exposition. The inequality $p(G) \leq 2c(G)$ follows from

$$p(G) = \sum_{v \in V} p(v) \leq \sum_{v \in V} \sum_{e \in \delta(v)} c(e) = 2 \sum_{e \in E} c(e) = 2c(G).$$

If T is a tree, root it at an arbitrary node r. Then $c(T) \leq p(T)$ since for each $v \neq r$, $p(v)$ is at least the cost of the parent edge of v.

We now show that $c(G) \leq \sqrt{|E|/2}p(G))$ It is sufficient to prove that

$$\sum_{xy \in E} \min\{p(x), p(y)\} \leq \sqrt{|E|/2} \sum_{v \in V} p(v) \tag{5}$$

for any graph $G = (V, E)$ with nonnegative weights $p(v)$ on the nodes. Suppose to the contrary that the statement is false, and let $G = (V, E)$ with p be a counterexample to (5) so that $\max_{v \in V} p(v) - \min_{v \in V} p(v)$ is minimal. Let $\mu = \min_{v \in V} p(v)$, let $U = \{v \in V : p(v) = \mu\}$, and let E_U be the set of edges in E with at least one endpoint in U. If $|E_U| \leq \sqrt{|E|/2}|U|$ then the statement is also false for $G' = (V', E') = (V - U, E - E_U)$ and p' being the restriction of p to V' since

$$\sum_{xy\in E'} \min\{p'(x),p'(y)\} \geq \sum_{xy\in E} \min\{p(x),p(y)\} - \sqrt{|E|/2}|U|\mu >$$
$$> \sqrt{|E|/2}\sum_{v\in V} p(v) - \sqrt{|E|/2}|U|\mu = \sqrt{|E|/2}\sum_{v\in V'} p'(v) >$$
$$> \sqrt{|E'|/2}\sum_{v\in V'} p'(v).$$

In particular, this implies a contradiction if $U = V$. Else, let $\mu' = \min\{p(v) : v \in V - U\}$ be the second minimum value of p. Then by setting $p(v) \leftarrow p(v) + \mu' - \mu$ for every $v \in U$ we obtain again a counterexample to (5). This contradicts our choice of G, p. □

We now prove Theorems 2 and 3. We need the following fundamental statement due to Mader.

Theorem 7 ([20]). *In a k-connected graph G, any cycle in which every edge is critical contains a node whose degree in G is k.*

Here an edge e of a k-connected graph G is *critical* (w.r.t. k-connectivity) if $G-e$ is not k-connected.

The following corollary (e.g., see [20]) is used to get a relation between $(k-1)$-edge covers and k-connected spanning subgraphs.

Corollary 2. *If $deg_J(v) \geq k-1$ for every node v of a graph J, and if F is an inclusion minimal edge set such that $J \cup F$ is k-connected, then F is a forest.*

Proof. If not, then F contains a cycle C of critical edges, but every node of this cycle is incident to 2 edges of C and to at least $k-1$ edges of G, contradicting Mader's Theorem. □

Proof of Theorem 3: By the assumption, we can find a subgraph J with $\deg_J(v) \geq k-1$ of power at most $p(J) \leq \beta\mathrm{opt}$. We reset the costs of edges in J to zero, and apply an α-approximation algorithm for the Min-Cost k-Connected Spanning Subgraph problem to compute an (inclusion) minimal edge set F so that $J+F$ is k-connected. By Corollary 2, F is a forest. Thus $p(F) \leq 2c(F) \leq 2\alpha\mathrm{opt}$, by Lemma 1. Combining, we get the desired statement.

The proof of the other direction is similar. We find a min-cost $(k-1)$-edge cover J in polynomial time, and reset the costs of its edges to zero. Then we use the ρ-approximation algorithm for MPk-CSS with the new cost function. The edges with nonzero cost in this new graph form a forest F, by Corollary 2. Then clearly $c(J)$ is at most the minimum cost of a k-connected spanning subgraph, and $c(F)$ is at most 2ρ times the minimum cost of a k-connected spanning subgraph, by Lemma 1. This gives a $(2\rho+1)$-approximation algorithm for the Min-Cost k-Connected Spanning Subgraph problem. □

We can combine the various existing approximation algorithms for the Min-Cost k-Connected Spanning Subgraph problem [7, 16, 17] to get better approximation for MPk-CSS. The currently best approximation ratios for the former are $O(\ln k \cdot \min\{\sqrt{k}, \frac{n+k}{n-k}\ln k\})$ [17] and $O(\ln k)$ for $n \geq 6k^2$ [7].

In particular, we set $\beta = k$ in Theorem 3 to get a $k(1 + o(1))$-approximation for any non-constant k. Using $\alpha = O(\ln k \cdot \min\{\sqrt{k}, \frac{n+k}{n-k} \ln k\})$ from [17] gives the bound in Theorem 2.

Remark: In [16] a $(2 + k/n)$-approximation was given for k-CS with metric costs. This does not imply that for metric costs we can set $\alpha = 2 + k/n$ in Theorem 3. Note that our algorithm first resets the costs of the edges in a k-edge cover to zero, and thus when applying an algorithm for min-cost k-CS the triangle inequality property *does not* hold for the obtained k-CS instance.

To prove Theorem 4, we combine Lemma 1 with the following theorem due to Cheriyan and Thurimella [6], which is the edge-connectivity counterpart of Corollary 2.

Theorem 8 ([6]). *If $deg_J(v) \geq k$ for every node v of a graph J, and if F is an inclusion minimal edge set such that $G \cup F$ is k-edge connected, then $|F| \leq n-1$.*

Proof of Theorem 4: We use the $O(\log^4)$-approximation for MPb-EC. Then we change the cost of these edges to zero and find the minimum cost k-edge connected subgraph using the known 2-approximation algorithms for the minimum cost k-edge connected subgraph problem [13]. From Lemma 1 and Theorem 8, the power of this augmentation is at most $2\sqrt{n/2}$ of the minimum power k-edge connected subgraph. This gives an $O(\sqrt{n})$-approximation algorithm. □

4 Proof of Theorem 5

To prove Theorem 5, we will show that approximating MPk-EDP is at least as hard as approximating the following problem, which is an alternative formulation, of the LabelCover-Max Problem defined in [15].

The MaxRep Problem:
Instance: A bipartite graph $H = (A \cup B, I)$, and equitable partitions $\mathcal{A}$ of A and $\mathcal{B}$ of B into q sets of same size each.
Objective: Choose $A' \subseteq A$ and $B' \subseteq B$ with $|A' \cap A_i| = |B' \cap B_j| = 1$ for each $i, j = 1, \ldots, q$ such that the subgraph induced by $A' \cup B'$ has maximum number of edges.

The bipartite graph and the partition of A and B induce a super-graph Γ in the following way: The vertices in Γ are the sets A_i and B_j. Two sets A_i and B_j are connected by a (super) edge in Γ if and only if there exist $a_i \in A_i$ and $b_j \in B_j$ which are adjacent in G. For our purposes, it is convenient (and possible) to assume that the graph Γ is regular. Say that every vertex in Γ has degree d, and hence, the number of super-edges is $h = qd$. Raz [23] proved:

Theorem 9. [23] *Let I be an instance of any NP-complete problem. For any $0 < \epsilon < 1$, there exists a (quasi-polynomial) reduction that maps I to an instance G of* MaxRep *with n vertices so that: 1) If I corresponds to a yes instance then there exists a feasible solution covering all super-edges, and 2) If I corresponds to a no instance, then every* MaxRep *feasible solution covers at most $h/2^{\log^{1-\epsilon} n}$ super-edges.*

In the above reduction the size n of the MaxRep instance G is quasi-polynomial in the size of the NP-complete instance. The following is implied from Theorem 9.

Theorem 10. *Unless NP* $\subseteq DTIME(n^{\text{polylog}n})$*, the* MaxRep *Problem admits no* $2^{\log^{1-\epsilon} n}$*-approximation algorithm, for any constant* $\epsilon > 0$*.*

The Reduction. We reduce MaxRep to MPk-EDP. Let H be the bipartite instance of MaxRep. Form an instance G for MPk-EDP as follows. First we put H into G and give all the edges of H directions from the A_i vertices to the B_j vertices. The edges of H are assigned cost n^3. Add a source s and a sink t. For each set A_i (B_i), $1 \leq i \leq q$, we also add a local source s_i (a local sink t_i). We add d edge-disjoint paths of length 2 from s_i, $1 \leq i \leq q$, into every $a_i^j \in A_i$, $1 \leq j \leq N$, and d edge-disjoint paths of length 2 from s to every s_i. These edges are given cost 0. Finally, we add d edge-disjoint paths of length 2 from every $b_i^j \in B_i$, $1 \leq i \leq q, 1 \leq j \leq N$ into t_i, and d edge-disjoint paths of length 2 from t_i into t. The first edge in every one of paths from a vertex in B_i to t_i gets cost n^3 while the rest of edges get cost 0.

A direct inspection shows that there exists $h = dq$ edge-disjoint paths from s to t and indeed we pick $k = h$ for the MPk-EDP instance.

Let H be a MaxRep instance resulting from a yes instance of the NP-complete instance and let G be the resulting MPk-EDP instance.

Lemma 4. *The graph* G *admits a subgraph* G' *of power-cost* $2qn^3$ *so that in* G' *there exist* $k = h$ *edge-disjoint paths from* s *to* t*.*

Proof. We select the following edges as a solution F to MPk-EDP. Let $a_i \in A_i$, $b_j \in B_j$ be a MaxRep solution covering all the superedges as guaranteed in Theorem 9. Add all the a_i to b_j edges into the solution. Note that the edge (a_i, b_j) exists as the chosen representatives cover all the super-edges. Include all the edges which are on a path from s to a_i, $1 \leq i \leq q$, and all edges which are on a path from b_j, $1 \leq j \leq q$, to t. Clearly the solution F admits h edge-disjoint $s-t$ paths. The solution pays n^3 per every a_i because of the A_i to B_j edges and n^3 per every b_j because of the d paths to t_j. □

Lemma 5. *If* G *corresponds to a no instance of* MaxRep *then the cost of any* MPk-EDP *solution is at least* $0.4qn^3 2^{\frac{\log^{1-\epsilon} n}{4}}$*.*

Proof. The idea of the proof is to start with a solution for MPk-EDP and use it to build a MaxRep solution that covers a number of superedges which is related to the cost of this solution. Let F be the solution to MPk-EDP. Call a vertex v *active* (with respect to F) if at least one edge in F touches v. Let A'_i (respectively, B'_j) be the collection of active vertices in A_i (respectively, B_j).

We may clearly assume that the outdegree of A'_i and B'_j vertices is nonzero (vertices that do not obey this can be discarded). The power-cost is thus at least $(\sum_i |A'_i| + \sum_j |B'|_j)n^3$.

Let $(\sum_i |A'_i| + \sum_j |B'|_j) = 2q\rho$. The average size of A'_i (respectively, $|B'_j|$) is at most 2ρ. Call an A_i *sparse* if $|A'_i| > 8\rho$. Similarly, B_j is *sparse* if $|B'_j| > 8\rho$.

Remove from the super-graph Γ all the sparse sets A_i and B_j. Clearly, the number of sparse A_i sets is no larger than $q/4$ and the same holds for B_j. Now we update the number of $s-t$ paths discarding paths of sparse sets. The loss of paths incurred by the removal of a sparse A_i or sparse B_j is at most d. Hence, the removal of sparse A_i and B_j sets incurs a loss of at most $2q/4d = h/2$ paths. Hence, at least $h/2$ $s-t$ edge-disjoint paths still exist after this update.

We now dilute the path collections so that at most one path remains between every pair of sets A_i, B_j. Since the remaining sets A_i and B_j are not sparse, the number of active vertices in each set is bounded by 8ρ. Hence, the total number of paths between every pair of sets A_i and B_j is at most $(8\rho)^2$. Therefore, the dilution results in a total number of paths of at least $\frac{h}{128\rho^2}$. Let F' be the subset of $\bigcup A'_i \cup \bigcup B'_j$ restricted to the non-sparse A_i, B_j.

We now create a feasible MaxRep solution by drawing a single vertex in every non-sparse A'_i and B'_i with all elements being equally likely to be chosen. Let F'' be the resulting set of unique representatives; Clearly F'' is a feasible MaxRep solution. Observe that a super-edge covered by F' has probability exactly $1/64\rho^2$ to be covered by F''. The expected number of superedges covered by F'' is $\frac{h}{8192\rho^4}$. This implies the existence of a MaxRep solution that covers this many superedges. By Theorem 9, $8192\rho^4 \geq 2^{\log^{1-\epsilon} n}$. Finally, we note that the probabilistic construction of F'' can be easily de-randomized using the method of conditional expectation and thus the claim follows. □

By Lemma 4 and 5, it is hard to approximate MPk-EDP within $2^{\frac{\log^{1-\epsilon} n}{4}}/10$. Since ϵ can be chosen to be any arbitrary constant, the hardness result follows.

References

1. E. Althaus, G. Calinescu, I. Mandoiu, S. Prasad, N. Tchervenski, and A. Zelikovsky, *Power efficient range assignment in ad-hoc wireless networks*, in Proceedings of IEEE Wireless Communications and Networking Conference (WCNC), 2003, pp. 1889–1894.
2. M. Bahramgiri, M. Hajiaghayi, and V. Mirrokni, *Fault-tolerant and 3-dimensional distributed topology control algorithms wireless multi-hop networks*, in Proceedings of the 11th IEEE International Conference on Computer Communications and Networks (ICCCN), IEEE Press, 2002, pp. 392–398.
3. D. M. Blough, M. Leoncini, G. Resta, and P. Santi, *On the symmetric range assignment problem in wireless ad hoc networks*, in TCS '02: Proceedings of the IFIP 17th World Computer Congress - TC1 Stream / 2nd IFIP International Conference on Theoretical Computer Science, Kluwer, B.V., 2002, pp. 71–82.
4. G. Calinescu, I. I. Mandoiu, and A. Zelikovsky, *Symmetric connectivity with minimum power consumption in radio networks*, in TCS '02: Proceedings of the IFIP 17th World Computer Congress - TC1 Stream / 2nd IFIP International Conference on Theoretical Computer Science, Kluwer, B.V., 2002, pp. 119–130.
5. G. Calinescu and P.-J. Wan, *Range assignment for high connectivity in wireless ad hoc network*, in Adhoc-Now, 2003, pp. 235–246.
6. J. Cheriyan and R. Thurimella, *Fast algorithms for k-shredders and k-node connectivity augmentation*, J. Algorithms, 33 (1999), pp. 15–50.

7. J. Cheriyan, S. Vempala, and A. Vetta, *An approximation algorithm for the minimum-cost k-vertex connected subgraph*, SIAM J. Comput., 32 (2003), pp. 1050–1055 (electronic).
8. A. Clementi, G. Huiban, P. Penna, G. Rossi, and Y. Verhoeven, *Some recent theoretical advances and open questions on energy consumption in ad-hoc wireless networks*, 2002, pp. 23–38.
9. A. E. F. Clementi, P. Penna, and R. Silvestri, *Hardness results for the power range assignmet problem in packet radio networks*, in RANDOM-APPROX '99: Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, Springer-Verlag, 1999, pp. 197–208.
10. A. Frank and É. Tardos, *An application of submodular flows*, Linear Algebra Appl., 114/115 (1989), pp. 329–348.
11. M. Hajiaghayi, N. Immorlica, and V. S. Mirrokni, *Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks*, in MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking, ACM Press, 2003, pp. 300–312.
12. S. Khuller and B. Raghavachari, *Improved approximation algorithms for uniform connectivity problems*, J. Algorithms, 21 (1996), pp. 434–450.
13. S. Khuller and U. Vishkin, *Biconnectivity approximations and graph carvings*, J. Assoc. Comput. Mach., 41 (1994), pp. 214–235.
14. L. M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc, *Power consumption in packet radio networks*, Theoret. Comput. Sci., 243 (2000), pp. 289–305.
15. G. Kortsarz, *On the hardness of approximating spanners*, Algorithmica, 30 (2001), pp. 432–450.
16. G. Kortsarz and Z. Nutov, *Approximating node connectivity problems via set covers*, Algorithmica, 37 (2003), pp. 75–92.
17. G. Kortsarz and Z. Nutov, *Approximation algorithm for k-node connected subgraphs via critical graphs*, in STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, ACM Press, 2004, pp. 138–145.
18. L. Li, J. Y. Halpern, P. Bahl, Y.-M. Wang, and R. Wattenhofer, *Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks*, in PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing, ACM Press, 2001, pp. 264–273.
19. E. L. Lloyd, R. Liu, M. V. Marathe, R. Ramanathan, and S. S. Ravi, *Algorithmic aspects of topology control problems for ad hoc networks*, Mob. Netw. Appl., 10 (2005), pp. 19–34.
20. W. Mader, *Ecken vom grad n in minimalen -fach zusammenhangenden Graphen*, Arch. Math. (Basel), 23 (1972), pp. 219–224.
21. J. B. Orlin, *A faster strongly polynomial minimum cost flow algorithm*, Oper. Res., 41 (1993), pp. 338–350.
22. C. H. Papadimitriou and M. Yannakakis, *Optimization, approximation, and complexity classes*, J. Comput. System Sci., 43 (1991), pp. 425–440.
23. R. Raz, *A parallel repetition theorem*, SIAM J. Comput., 27 (1998), pp. 763–803 (electronic).
24. V. Rodoplu and T. H. Meng, *Minimum energy mobile wireless networks*, IEEE J. Selected Areas in Communications, 17 (1999), pp. 1333–1344.
25. R. Wattenhofer, L. Li, V. Bahl, and Y. Wang, *Distributed topology control for power efficient operation in multihop wireless ad hoc networks*, in Proceedings of twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), IEEE Press, 2001, pp. 1388–1397.

Packing Steiner Forests

(Extended Abstract)

Lap Chi Lau*

Department of Computer Science,
University of Toronto,
Toronto, ON, Canada M5S 3G4
chi@cs.toronto.edu

Abstract. Given an undirected multigraph G and a set $\mathcal{S} := \{S_1, \ldots, S_t\}$ of disjoint subsets of vertices of G, a Steiner $\mathcal{S}$-forest F is an acyclic subgraph of G such that each S_i is connected in F for $1 \leq i \leq$. In this paper, we study the Steiner Forest Packing problem where we seek a largest collection of edge-disjoint $\mathcal{S}$-forests. The main result is a connectivity-type sufficient condition for the existence of k edge-disjoint $\mathcal{S}$-forest, that yields the first polynomial time approximation algorithm for the Steiner Forest Packing problem. We end this paper by a conjecture in a more general setting.

1 Introduction

Given an undirected multigraph G and a set $\mathcal{S} := \{S_1, \ldots, S_t\}$ of disjoint subsets of vertices of G, a Steiner $\mathcal{S}$-forest F is an acyclic subgraph of G such that each S_i is connected in F for $1 \leq i \leq t$. The Steiner Forest Packing problem is to find a largest collection of edge-disjoint $\mathcal{S}$-forests. This is a generalization of the Steiner Tree Packing problem where we are given G and a subset of vertices $S \subseteq V(G)$, and the goal is to find a largest collection of edge-disjoint trees that each connects S. The Steiner Tree Packing problem is well-studied in the literature as it generalizes the edge-disjoint s,t-paths problem [11] as well as the spanning tree packing problem [14, 12], and also it has applications in network broadcasting and VLSI circuit design. We say a set $S \subseteq V(G)$ is k-edge-connected in G if there are k edge-disjoint paths between u, v for all $u, v \in S$, and a tree is a S-tree if it connects S. A necessary condition for G to have k edge-disjoint S-trees is that S is k-edge-connected in G. The following conjecture by Kriesell [7] gives a sufficient condition for the existence of k edge-disjoint S-trees:

> **Kriesell conjecture:** If S is $2k$-edge-connected in G, then G has k edge-disjoint S-trees.

* Part of this work was done when the author visited the Operations Research Department of the Eötvös Loránd University. The visit was supported by European MCRTN Adonet, Contract No. 504438.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 362–376, 2005.

This beautiful conjecture is the best possible and has generated much interest [13, 3, 6, 7, 2, 8, 9], recently it is shown in [9] that the conjecture is true if $2k$ is replaced by $26k$.

A related problem is the Minimum Steiner Forest problem, where the goal is to find a minimum-cost $\mathcal{S}$-forest F in G. Goemans and Williamson [4] gave a primal-dual 2-approximation algorithm for the Minimum Steiner Forest problem. Very recently, Chekuri and Shepherd [1] present a new 2-approximation algorithm for the Minimum Steiner Forest problem via the Steiner Forest Packing problem. Specifically, they show that given an *Eulerian graph* G, if each S_i is $2k$-edge-connected in G, then there are k edge-disjoint $\mathcal{S}$-forests in G. They then show that this result implies the integrality gap of the standard LP formulation of the Minimum Steiner Forest problem is at most 2.

Motivated by the above results, we study the Steiner Forest Packing problem in general graphs. The main result is the following:

Theorem 1. *Given an undirected multigraph G and a set $\mathcal{S} := \{S_1, \ldots, S_t\}$ of disjoint subsets of vertices of G. If each S_i is Qk-edge-connected in G, then there are k edge-disjoint $\mathcal{S}$-forests in G.*

Remark: The best upper bound on Q that we can achieve is 32; however, the proof is fairly involved. In this extended abstract, for the sake of clarity, we only give a proof for $Q = 56$ which contains most of the underlying ideas. We also remark that the proof of Theorem 1 is constructive, and this yields the first polynomial time constant factor approximation algorithm for the Steiner Forest Packing problem.

2 Preliminaries

The organization of this section is as follows. We first develop the necessary notations and concepts in Section 2.1. Then, we review the basic theorems that we will be using in Section 2.2. Finally, we give an overview of the proof of our result in Section 2.3.

2.1 Basic Definitions

Given an undirected multigraph G and a set $\mathcal{S} := \{S_1, \ldots, S_t\}$ of disjoint subsets of vertices of G. We call each S_i a **group**, note that we can assume that each group is of size at least 2. A subgraph H of G is a **$\mathcal{S}$-subgraph** if each S_i is connected in H for $1 \leq i \leq t$; a subgraph H is a **double $\mathcal{S}$-subgraph** of G if H is a $\mathcal{S}$-subgraph of G and every vertex in $S_1 \cup \ldots \cup S_t$ is of degree at least 2 in H. Given $S \subseteq V(G)$, we say a subgraph H of G is a **S-subgraph** if H is *connected* and S is connected in H; a subgraph H is a **double S-subgraph** of G if H is a S-subgraph of G and every vertex in S is of degree at least 2 in H. A subgraph H **spans** a subset of vertices U if $U \subseteq V(H)$.

In the rest of this paper, we denote $S := S_1 \cup \ldots \cup S_t$. A subset of vertices X is a **$\mathcal{S}$-separating cut** if (i) $S \cap X \neq \emptyset$, $S \cap (V(G) - X) \neq \emptyset$ and (ii) for

each S_i, either $S_i \subseteq X$ or $S_i \subseteq V(G) - X$. We denote by $\delta_G(X)$ the set of edges in G with one endpoint in X and the other endpoint in $V(G) - X$, the open neighbourhood set of a vertex v in G by $N_G(v)$, and the induced subgraph on a set of vertices X by $G[X]$. A **core** C is a $\mathcal{S}$-separating cut with $\delta_G(C) \leq Qk$ and $|C|$ minimal. Let R be a specified set of vertices such that $R \cap S = \emptyset$ and each vertex in R is of degree at least Qk. A subset of vertices X is a R**-isolating cut** if $R \cap X \neq \emptyset$ and $S \cap X = \emptyset$.

Given a vertex v, we denote by $E(v)$ the set of edges with an endpoint in v. $\mathcal{P}_k(v) := \{E_1(v), \ldots, E_k(v)\}$ is a **balanced edge-subpartition** of $E(v)$ if (i) $E_1(v) \cup E_2(v) \cup \ldots \cup E_k(v) \subseteq E(v)$, (ii) $|E_i(v)| \geq 2$ for $1 \leq i \leq k$ and (iii) $E_i(v) \cap E_j(v) = \emptyset$ for $i \neq j$. We denote the set of neighbours of u in $E_i(u)$ by $N_{E_i}(u)$. Given k edge-disjoint subgraphs $\{H_1, \ldots, H_k\}$ of G, a vertex v is called **balanced-extendible** with respect to $\{H_1, \ldots, H_k\}$ (or just balanced-extendible if the context is clear) if there exists a balanced edge-subpartition $\{E_1(v), \ldots, E_k(v)\}$ of $E(v)$ so that $H_i \cap E(v) \subseteq E_i(v)$. An easy fact that will be used many times later: if there are $2k$ edges in $E(v)$ not used in any of $\{H_1, \ldots, H_k\}$, then v is balanced-extendible (with respect to $\{H_1, \ldots, H_k\}$). And, of course, if v is of degree at least 2 in each H_i, then v is balanced-extendible.

Cut Decomposition: Given a multigraph G a subset of vertices $Y \subset V(G)$, the cut decomposition operation constructs two multigraphs G_1 and G_2 from G as follows. G_1 is obtained from G by contracting $V(G) - Y$ to a single vertex v_1, and keeping all edges from Y to v_1 (even if this produce multiple edges). Similarly, G_2 is obtained from G by contracting Y to a single vertex v_2, and keeping all edges from $V(G) - Y$ to v_2. So, $V(G_1) = Y \cup \{v_1\}, \delta_G(Y) \subseteq E(G_1)$ and $V(G_2) = (V(G) - Y) \cup \{v_2\}, \delta_G(Y) \subseteq E(G_2)$. Notice that for each edge $e \in \delta_G(Y)$, e will appear in both G_1 and G_2 (i.e. e in G_1 is incident with v_1 where e in G_2 is incident with v_2). So, given an edge e incident with v_1 in G_1, we refer to the same edge in G_2 incident with v_2 the **corresponding edge** of e in G_2, and vice versa. The cut decomposition operation will be used several times later, the following are two basic properties of G_1 and G_2:

1. For each pair of vertices u, v in G_i, the number of edge-disjoint paths between u, v in G_i is at least the number of edge-disjoint paths between u, v in G. In particular, if a set S is Qk-edge-connected in G, then $S \cap V(G_1)$ and $S \cap V(G_2)$ are Qk-edge-connected in G_1 and G_2 respectively.
2. The degree of each vertex v in $V(G_i) - \{v_i\}$ is equal to the degree of v in G.

2.2 Basic Tools

The following Mader's splitting-off lemma has been a standard tool in many edge-connectivity related problems. Let G be a graph and $e_1 = xy, e_2 = yz$ be two edges of G, the operation of obtaining $G(e_1, e_2)$ from G by replacing $\{e_1, e_2\}$ by a new edge $e' = xz$ is called **splitting off at** y. The splitting-off operation is said to be **suitable**, if the number of edge-disjoint u, v-paths in $G(e_1, e_2)$ is the same as the number of edge-disjoint u, v-paths in G for $u, v \in V(G) - y$.

Lemma 1. (Mader's Splitting Lemma) *[10] Let x be a vertex of a graph G. Suppose that x is not a cut vertex and that x is incident with at least 4 edges and adjacent to at least 2 vertices. Then there exists a suitable splitting-off operation of G at x.*

By repeatedly applying Mader's splitting-off lemma, one can make the following assumption on the degree of vertices in $V(G) - S$. The proof of Lemma 2 is now standard and appeared in [3, 6, 7, 9], we omit the details here.

Lemma 2. *Every vertex in $V(G) - S$ is incident with exactly three edges and adjacent to exactly three vertices.*

The following are two results on the Steiner Tree Packing problem which we will be using. Somewhat surprisingly, to prove the result in this paper, we could not use Theorem 3 as a black box. In fact, we need to revise and generalize many parts of the proof of Theorem 3 in order to meet our requirements, this point will be elaborated more in the next subsection.

Theorem 2. *[3] If there is no edge between any two vertices in $V(G) - S$ and S is $3k$-edge-connected in G, then G has k edge-disjoint S-trees.*

Theorem 3. *[9] If S is $26k$-edge-connected in G, then G has k edge-disjoint S-trees.*

2.3 Overview

Here we outline the approach of Chekuri and Shepherd [1]. In [1], they consider the Steiner Forest Packing problem when G is Eulerian. This allows them to assume that $V(G) = S$ (recall that $S := S_1 \cup \ldots \cup S_t$) by repeatedly applying the Mader's splitting-off lemma. Then they find a core C in G and show that $G[C]$ has k edge-disjoint spanning trees $T_1, \ldots, T_k$, by using Tutte [14] and Nash-Williams [12] result on spanning tree packing. Now they contract C in G and obtain a new graph G^*. Note that C contains a group and thus has at least two vertices, so G^* has fewer vertices than G. By induction, G^* has k edge-disjoint Steiner forests $F_1, F_2, \ldots, F_k$. Now, as each tree is *spanning* in C, $F_1 \cup T_1, \ldots, F_k \cup T_k$ are the desired k edge-disjoint Steiner forests in G.

However, if G is non-Eulerian, we cannot assume that $V(G) = S$. So, even if we assume the existence of a core C of G so that $G[C]$ has k edge-disjoint Steiner trees $T_1, \ldots, T_k$, and also the existence of k edge-disjoint Steiner forests $F_1, \ldots, F_k$ of G^* as constructed above, the main difficulty is that we cannot guarantee that for all pair of vertices which are connected in F_i are still connected in $F_i \cup T_i$. Here the extension property introduced in [9] comes into the picture. Roughly, we show that there are k edge-disjoint Steiner trees in $G[C]$ that "extend" $F_1, \ldots, F_k$ so that $F_1 \cup T_1, \ldots, F_k \cup T_k$ are actually k edge-disjoint Steiner forests. Unlike the situation in the Steiner Tree Packing problem, however, we also need to prove structural properties on $F_1, \ldots, F_k$ so that they can actually be extended (not every $F_1, \ldots, F_k$ can be extended). This requires

us to revise and generalize the extension theorem in [9], in particular we need to add an additional constraint on the extension theorem, which causes the constant in Theorem 1 (i.e. approximation ratio) to be slightly bigger than that in Theorem 3.

The organization of the proof is as follows. In Section 3, we prove structural results on R-isolating cut, $\mathcal{S}$-separating cut and core. With these, in Section 4, we show how to reduce a stronger version (see Theorem 4) of Theorem 1 to the Steiner Tree Packing problem with the extension property (defined in subsection 4.1) and an additional requirement (see Theorem 5). Then, in Section 5, we prove the extension theorem (Theorem 5).

3 Structural Results

This section contains structural results on R-isolating cut and $\mathcal{S}$-separating cut; these results will be used in Section 4 to reduce the Steiner Forest Packing problem to a modified version of the Steiner Tree Packing problem (see Theorem 5).

3.1 R-Isolating Cut

Let $\mathcal{G}$ be a minimal counterexample of Theorem 1 (or Theorem 4) in the following.

Lemma 3. *$\mathcal{G}$ has no R-isolating cut Y with $|\delta_{\mathcal{G}}(Y)| \leq (Q-2)k$.*

Proof. Suppose not. Consider a R-isolating cut Y with $|\delta_{\mathcal{G}}(Y)|$ minimum. Apply the cut decomposition operation on $\mathcal{G}$ and Y to obtain two graphs G_1 and G_2, let $R_1 := R \cap V(G_1)$ and $R_2 := R \cap V(G_2)$. Also, by definition of Y, $S \subseteq V(G_2)$. From the properties of the cut decomposition operation, S is Qk-edge-connected in G_2 and every vertex of R_2 is of degree at least Qk in G_2. Note that $R_1 \neq \emptyset$, hence $|Y| \geq 2$ as each vertex in R is of degree at least Qk while $|\delta_{\mathcal{G}}(Y)| \leq (Q-2)k < Qk$. As $|Y| \geq 2$, G_2 is smaller than $\mathcal{G}$. By the choice of $\mathcal{G}$, G_2 has k edge-disjoint $\mathcal{S}$-subgraphs $\{H'_1, \ldots, H'_k\}$ such that every vertex in R_2 is balanced-extendible.

Let $l := |\delta_{\mathcal{G}}(Y)|$, then $l \leq (Q-2)k$. We claim that each vertex $r \in R_1$ has l edge-disjoint paths to v_1 in G_1. Suppose not, then there exists $Y' \subseteq V(G_1)$ so that $r \in Y'$, $v_1 \notin Y'$ and $|\delta_{G_1}(Y')| < l$. Hence, Y' is a R-isolating cut in $\mathcal{G}$ with $|\delta_{\mathcal{G}}(Y')| < l$, but this contradicts the minimality of $|\delta_{\mathcal{G}}(Y)|$. Therefore, each vertex in R_1 has l edge-disjoint paths to v_1. Choose a vertex $r^* \in R_1$ so that the total length of the l edge-disjoint paths $\{P_1, \ldots, P_l\}$ from v_1 to r^* is minimized. Let $P := \{P_1, \ldots, P_l\}$ and $H := P_1 \cup P_2 \cup \ldots \cup P_l$. We claim that r is of degree at most l in H for each $r \in R_1$. Suppose not, let $r \in R_1$ be of degree at least $l+1$ in H. Then $r \neq r^*$ and at least $(l+1)/2$ paths from v_1 to r^* pass through r, say $\{P_1, \ldots, P_{(l+1)/2}\}$. For $1 \leq i \leq (l+1)/2$, let the subpath of P_i from v_1 to r be P'_i, and the subpath of P_i from r^* to r be P''_i. Then we claim that $H - P''_{(l+1)/2}$ contains l edge-disjoint paths from v_1 to r;

indeed, $\{P'_1, \ldots, P'_{(l+1)/2}, P_{(l+1)/2+1} \cup P''_1, \ldots, P_l \cup P''_{(l-1)/2}\}$ are l edge-disjoint paths from v_1 to r in $H - P''_{(l+1)/2}$. This contradicts with the choice of r^* and thus r is of degree at most $l \leq (Q-2)k$ for each $r \in R_1$ in H. In particular, each $r \in R_1$ is balanced-extendible with respect to $\{P_1, \ldots, P_l\}$.

Now, we show that the k edge-disjoint double $\mathcal{S}$-subgraphs $\{H'_1, \ldots, H'_k\}$ in G_2 can be extended to k edge-disjoint double $\mathcal{S}$-subgraphs in $\mathcal{G}$, by using the edge-disjoint paths from v_1 to r^* in G_1 constructed in the previous paragraph. For each H'_i in G_2, let $E_i(v_2) := E(H'_i) \cap E(v_2)$ and $E_i(v_1)$ be the corresponding edge set in G_1. Let P_i be the paths in P of G_1 using the corresponding edges in $E_i(v_1)$ (note that $|P_i| = |E_i(v_1)|$), and we set $H_i := H'_i \cup P_i$ to be a subgraph of $\mathcal{G}$. Now each path in H'_i using v_2 in G_2 is still connected in $\mathcal{G}$ via the paths of P_i in G_1. Since H'_i is a double $\mathcal{S}$-subgraph in G_2, H_i is a double $\mathcal{S}$-subgraph in $\mathcal{G}$. Also, since H'_i and H'_j are edge-disjoint in G_2 for $i \neq j$, and P_i and P_j are edge-disjoint in G_1 for $i \neq j$, H_i and H_j are edge-disjoint in $\mathcal{G}$ for $i \neq j$. Finally, since vertices in R_1 and R_2 are balanced-extendible in G_1 and G_2 respectively, vertices in R are balanced-extendible in $\mathcal{G}$ with respect to $\{H_1, \ldots, H_k\}$. As a result, $\{H_1, \ldots, H_k\}$ are k edge-disjoint double $\mathcal{S}$-subgraphs in $\mathcal{G}$ so that every vertex in R is balanced-extendible. This contradiction completes the proof of the lemma. □

3.2 $\mathcal{S}$-Separating Cut and Core

Let C be a core of $\mathcal{G}$. We apply the cut decomposition operation on $\mathcal{G}$ and C to obtain two graphs G_1 and G_2, and let $C \subseteq V(G_1)$, $R_1 := R \cap V(G_1)$ and $S_1 := S \cap V(G_1)$ (recall that $S := S_1 \cup \ldots \cup S_t$). We prove some structural results of G_1 in the following lemma.

Lemma 4. *Let G_1 be defined above. Then S_1 is Qk-edge-connected in G_1 and $S_1 \cup R_1$ is $(Q-2)k$-edge-connected in G_1. Besides, either one of the following must be true:*

1. *$S_1 \cup \{v_1\}$ is Qk-edge-connected in G_1.*
2. *$N_{G_1}(v_1) \subseteq S_1 \cup R_1$.*

Proof. First we prove that S_1 is Qk-edge-connected in G_1 (note that S_1 maybe the union of several groups). Suppose not, then there exists $Y \subseteq V(G_1)$ such that $Y \cap S_1 \neq \emptyset$, $(V(G_1) - Y) \cap S_1 \neq \emptyset$, and $|\delta_{G_1}(Y)| < Qk$. Without loss of generality, we can assume that $v_1 \notin Y$ and hence Y is also a cut in $\mathcal{G}$. Since $|\delta_{G_1}(Y)| < Qk$, each group in G_1 is either contained in Y or disjoint from Y; otherwise this contradicts with the assumption that each group is Qk-edge-connected in $\mathcal{G}$. So, Y is a $\mathcal{S}$-separating cut in $\mathcal{G}$ with $|\delta_{\mathcal{G}}(Y)| < Qk$ and $|Y| < |C|$ (as $(V(G_1) - Y) \cap S_1 \neq \emptyset$). This contradicts with the fact that C is a core, and so S_1 is Qk-edge-connected in G_1.

Next we prove that $S_1 \cup R_1$ is $(Q-2)k$-edge-connected in G_1. Suppose not, then there exists $Y \subseteq V(G_1)$ such that $Y \cap (R_1 \cup S_1) \neq \emptyset$, $(V(G_1) - Y) \cap (R_1 \cup S_1) \neq \emptyset$, and $|\delta_{G_1}(Y)| < (Q-2)k$. Since S_1 is Qk-edge-connected in G_1,

either $S_1 \subseteq Y$ or $S_1 \subseteq V(G_1) - Y$. Without loss of generality, we assume that $v_1 \notin Y$ and hence Y is also a cut in $\mathcal{G}$. If $S_1 \subseteq Y$, by a similar argument as in the previous paragraph, Y is a $\mathcal{S}$-separating cut in $\mathcal{G}$ with $|\delta_{\mathcal{G}}(Y)| < Qk$ and $|Y| < |C|$ (as $(V(G_1) - Y) \cap R_1 \neq \emptyset$), and this contradicts with the fact that C is a core. So $S_1 \subseteq V(G_1) - Y$. This, however, implies that Y is a R-isolating cut in $\mathcal{G}$ with $|\delta_{\mathcal{G}}(Y)| < (Q-2)k$, which contradicts Lemma 3. As a result, $S_1 \cup R_1$ is $(Q-2)k$-edge-connected in G_1.

Finally, we prove that if $N_{G_1}(v_1) \not\subseteq S_1 \cup R_1$, then $S_1 \cup \{v_1\}$ is Qk-edge-connected in G_1. First, we show that v_1 must be of degree Qk. Suppose not, then v_1 is a vertex of degree less than Qk. Let $w \in N_{G_1}(w_1)$ be a vertex in $V(G_1) - S_1 - R_1$. Since w is of degree 3 by Lemma 2, $|\delta_{\mathcal{G}}(C - w)| \leq Qk$ and hence $C - w$ is also a $\mathcal{S}$-separating cut which contradicts the fact that C is a core. So, v_1 is of degree exactly Qk. Suppose indirectly that $S_1 \cup \{v_1\}$ is not Qk-edge-connected in G_1, then there exists $Y \subseteq V(G_1)$ such that $v_1 \notin Y$, $Y \cap S_1 \neq \emptyset$ and $|\delta_{G_1}(Y)| < Qk$. Since S_1 is Qk-edge-connected in G_1, we have $S_1 \subseteq Y$. Also, since v_1 is of degree Qk but $|\delta_{G_1}(Y)| < Qk$, we have $|V(G_1) - Y| \geq 2$ and hence $|Y| < |C|$. This implies that Y is also a $\mathcal{S}$-separating cut which contradicts with the fact that C is a core. Therefore, if $N_{G_1}(v_1) \not\subseteq S_1 \cup R_1$, then $S_1 \cup \{v_1\}$ is Qk-edge-connected in G_1. This completes the proof of the lemma. □

4 Proof of the Main Theorem

We are going to prove the following stronger theorem, which helps us to reduce the Steiner Forest Packing problem to a modified version of the Steiner Tree Packing problem (see Theorem 5).

Theorem 4. *Given G, $\mathcal{S} := \{S_1, \ldots, S_t\}$, $R \subseteq V(G)$. If each S_i is Qk-edge-connected in G and each vertex in R is of degree at least Qk, then there are k edge-disjoint double $\mathcal{S}$-subgraphs $\{H_1, \ldots, H_k\}$ of G so that every vertex in R is balanced-extendible with respect to $\{H_1, \ldots, H_k\}$.*

Theorem 4 immediately implies Theorem 1 (just set $R := \emptyset$). In the following, we consider a minimal counterexample $\mathcal{G}$ of Theorem 4, and show that it does not exist.

4.1 The Extension Theorem

The extension property defined below is crucial in applying divide-and-conquer strategy to decompose the original problem instance to smaller instances with restricted structures.

Definition 1. (The Extension Property) *Given G, $S \subseteq V(G)$, and an edge-subpartition $\mathcal{P}_k(v) := \{E_1(v), \ldots, E_k(v)\}$ of a vertex v. $\{H_1, \ldots, H_k\}$ are k edge-disjoint double S-subgraphs that* **extend** *$\mathcal{P}_k(v)$ if for $1 \leq i \leq k$:*

(1) $E_i(v) \subseteq E(H_i)$;
(2) $H_i - v$ is a double $(S - v)$-subgraph that spans $N_{E_i}(v)$.

The following lemma illustrates how to use the cut decomposition operation to reduce the Steiner Tree Packing problem (with the extension property and the balanced-extendible constraint) in a graph to the same problem in two smaller graphs.

Lemma 5. *Given G, $S, R \subseteq V(G)$, $v \in S$ and an edge-subpartition $\mathcal{P}_k(v)$ of v. Let G_1 and G_2 be two graphs obtained from the cut decomposition operation of a graph G and assume $v \in G_2$, let $S_1 := S \cap V(G_1), S_2 := S \cap V(G_2), R_1 := R \cap V(G_1)$, and $R_2 := R \cap V(G_2)$. Suppose $\{H_1^2, \ldots, H_k^2\}$ are k edge-disjoint double S_2-subgraphs of G_2 that extend $\mathcal{P}_k(v)$ so that every vertex in R_2 is balanced-extendible. Let $\mathcal{P}_k(v_2) := \{H_1^2 \cap E(v_2), \ldots, H_k^2 \cap E(v_2)\}$ and $\mathcal{P}_k(v_1)$ be the corresponding edge-subpartition of v_1 in G_1. If $\{H_1^1, \ldots, H_k^1\}$ are k edge-disjoint double S_1-subgraphs of G_1 that extend $\mathcal{P}_k(v_1)$ so that every vertex in R_1 is balanced-extendible, then $\{H_1^1 \cup H_1^2, \ldots, H_k^1 \cup H_k^2\}$ are k edge-disjoint double S-subgraphs of G that extend $\mathcal{P}_k(v)$ so that every vertex in R is balanced-extendible.*

Proof. The statement of the lemma is somewhat long, but the proof is quite straightforward and a similar proof (without the balanced-extendible constraint) is available in Lemma 3.2 of [9]. We omit the proof here for the sake of space. □

Of course, not every edge-subpartition of an arbitrary vertex can be extended. The following extension theorem, which is at the heart of the proof of Theorem 4, gives a sufficient condition for an edge-subpartition of a vertex to be extendible. The proof of Theorem 5 is in Section 5.

Theorem 5. *Given G, $S, R \subseteq V(G)$. If S is Qk-edge-connected in G and $S \cup R$ is $(Q-2)k$-edge-connected in G, then there are k edge-disjoint double S-subgraphs in G such that every vertex in R is balanced-extendible. Furthermore, given a balanced edge-subpartition $\mathcal{P}_k(v)$ of a vertex $v \in S$ of degree Qk, then there are k edge-disjoint double S-subgraphs that extend $\mathcal{P}_k(v)$ such that every vertex in R is balanced-extendible.*

4.2 Proof of Theorem 4

Now, with Theorem 5, we are ready to prove Theorem 4.

Proof. First suppose that S is Qk-edge-connected in $\mathcal{G}$. Then, by Lemma 3, $S \cup R$ is $(Q-2)k$-edge-connected in $\mathcal{G}$. Hence, Theorem 5 (without using the extension property) implies the theorem in this case.

So S is not Qk-edge-connected in $\mathcal{G}$, then a core C exists. We apply the cut decomposition operation on $\mathcal{G}$ and C to obtain two graphs G_1 and G_2, and assume without loss of generality that $C \subset V(G_1)$. Let $S_1 := S \cap V(G_1), S_2 := S \cap V(G_2) R_1 := R \cap V(G_1), R_2 := R \cap V(G_2)$, and $\mathcal{S}_1$ and $\mathcal{S}_2$ be the groups contained in S_1 and S_2 respectively. By the properties of the cut decomposition operation, each group in $\mathcal{S}_1$ and $\mathcal{S}_2$ is Qk-edge-connected in G_1 and G_2 respectively. Also, vertices in R_1 and R_2 are of degree at least Qk in G_1 and G_2 respectively. By the choice of $\mathcal{G}$, there are k edge-disjoint double $\mathcal{S}_2$-subgraphs

$\{H_1^2, \ldots, H_k^2\}$ in G_2 such that every vertex in R_2' is balanced-extendible. Based on Lemma 4, we distinguish two cases based on the possible structures of v_1.

The first case is that $S_1 \cup \{v_1\}$ is Qk-edge-connected in G_1, let $S_1' := S_1 \cup \{v_1\}$ and $R_2' := R_2 \cup \{v_2\}$. Let $\mathcal{P}_k^2(v_2) := \{H_1^2 \cap E(v_2), \ldots, H_k^2 \cap E(v_2)\}$ be the edge-subpartition on v_2 induced by $\{H_1^2, \ldots, H_k^2\}$ in G_2, and $\mathcal{P}_k^1(v_1)$ be the corresponding edge-subpartition defined on $E(v_1)$ in G_1. Since $v_2 \in R_2'$, v_2 is balanced-extendible with respect to $\{H_1^2, \ldots, H_k^2\}$. Now, we apply Theorem 5 on G_1 with S_1', R_1 and $\mathcal{P}_k^1(v_1)$ to get k edge-disjoint double $\mathcal{S}_1'$-subgraphs $\{H_1^1, \ldots, H_k^1\}$ in G_1 that extend $\mathcal{P}_k^1(v_1)$, so that every vertex in R_1 is balanced-extendible. Then, by Lemma 5, we obtain k edge-disjoint double $\mathcal{S}$-subgraphs $\{H_1, \ldots, H_k\}$ in $\mathcal{G}$ such that every vertex in R is balanced-extendible.

The second case is that $N_{G_1}(v_1) \subseteq S_1 \cup R_1$. Consider $G_1 - v_1$. Since v_1 is of degree at most Qk, $S_1 \cup R_1$ is at least $(Q-2)k - Qk/2 \geq 26k$-edge-connected in G_1. By Theorem 3, G_1 has k edge-disjoint double $S_1 \cup R_1$-subgraphs $\{H_1', \ldots, H_k'\}$ in $G_1 - v_1$. Setting $H_i^1 := H_i' \cup E_i$, $\{H_1^1, \ldots, H_k^1\}$ are k edge-disjoint double $S_1 \cup R_1$-subgraphs in G_1 that extend $\mathcal{P}_k(v_1)$. Then, by Lemma 5, we obtain k edge-disjoint double $\mathcal{S}$-subgraphs $\{H_1, \ldots, H_k\}$ in $\mathcal{G}$ such that every vertex in R is balanced-extendible.

By Lemma 4, these are the only two cases that could happen in G_1. Therefore, $\mathcal{G}$ is not a counterexample and the theorem follows. □

5 Proof of the Extension Theorem

In this section, we will prove Theorem 5 by showing that a minimal counterexample $\mathcal{G}$ of Theorem 5 does not exist. In Section 5.1, we first prove that there is no edge between two vertices in $V(\mathcal{G}) - S - R$, which allow us to apply Theorem 2 to prove Theorem 5. Then, in Section 5.2, we construct an auxiliary graph G' from $\mathcal{G}$ and show some properties of G'. We then introduce the concept of diverging paths and common paths in Section 5.3. Finally, we distinguish two cases of the structure of G' and show that each is impossible in Section 5.4 and Section 5.5.

5.1 There Is no Edge Between Two Vertices in $V(\mathcal{G}) - S - R$

Lemma 6. *There is no edge in $\mathcal{G}$ with both endpoints in $V(\mathcal{G}) - S - R$.*

Proof. Suppose indirectly that e is such an edge. If S is Qk-edge-connected and $S \cup R$ is $(Q-2)k$-edge-connected in $\mathcal{G} - e$, then by the choice of $\mathcal{G}$, $\mathcal{G} - e$ satisfies Theorem 5 and thus $\mathcal{G}$. So we may assume that either (i) there exists $Y \subseteq V(\mathcal{G})$ and $e \in \delta_{\mathcal{G}}(Y)$ so that $Y \cap S \neq \emptyset$, $(V(\mathcal{G}) - Y) \cap S \neq \emptyset$ and $|\delta_{\mathcal{G}}(Y)| = Qk$; or (ii) there exists $Y \subseteq V(\mathcal{G})$ and $e \in \delta_{\mathcal{G}}(Y)$ so that $S \subseteq Y$, $(V(\mathcal{G}) - Y) \cap R \neq \emptyset$ and $|\delta_{\mathcal{G}}(Y)| = (Q-2)k$. In either case, we apply the cut decomposition operation on $\mathcal{G}$ and Y to obtain two graphs G_1 and G_2, and let $Y \subset G_1$, $S_1 := S \cap V(G_1)$, $R_1 := R \cap V(G_1)$, $S_2 := S \cap V(G_2)$ and $R_2 := R \cap V(G_2)$. Notice that since both G_1 and G_2 contain a vertex in $V(\mathcal{G}) - S - R$ (an endpoint of e), both G_1 and G_2 are smaller than $\mathcal{G}$ and hence they both satisfy Theorem 5.

We first consider case (i). In this case, S_1 and S_2 are non-empty. Hence, $S_1 \cup \{v_1\}$ and $S_2 \cup \{v_2\}$ are Qk-edge-connected in G_1 and G_2, respectively. We let $S_1' := S_1 \cup \{v_1\}$ and $S_2' := S_2 \cup \{v_2\}$. Without loss of generality, we assume that v is in G_2, where v is the vertex to be extended in $\mathcal{G}$. By the choice of $\mathcal{G}$, there are k edge-disjoint double S_2'-subgraphs $\{H_1^2, \ldots, H_k^2\}$ in G_2 that extend $\mathcal{P}_k(v)$ such that every vertex in R_2 is balanced-extendible. Since $v_2 \in S_2'$, $\mathcal{P}_k(v_2) := \{H_1^2 \cap E_{G_2}(v_2), \ldots, H_k^2 \cap E_{G_2}(v_2)\}$ is a balanced edge-subpartition of v_2. Let $\mathcal{P}_k(v_1)$ be the corresponding balanced edge-subpartition of v_1 in G_1. Again, by the choice of $\mathcal{G}$, there are k edge-disjoint double S_1'-subgraphs $\{H_1^1, \ldots, H_k^1\}$ in G_1 that extend $\mathcal{P}_k(v_1)$ such that every vertex in R_1 is balanced-extendible. Now, by applying Lemma 5, there are k edge-disjoint double S-subgraphs in $\mathcal{G}$ that extend $\mathcal{P}_k(v)$ such that every vertex in R is balanced-extendible.

Now consider case (ii). Notice that we must have that $v \in G_1$, since $v \in S$ and we assume $S \subseteq Y$ and $Y \subset V(G_1)$. Then $V(\mathcal{G}) - Y$ is a R-isolating cut which does not contain v in $\mathcal{G}$. By using exactly the same argument as in Lemma 3, we can show that this contradicts with the fact that $\mathcal{G}$ is a minimal counterexample. So case (ii) cannot happen either, and this completes the proof. □

5.2 Construction and Properties of G'

The case when $|S \cup R| = 2$ is trivial. Henceforth, we assume that $|S \cup R| \geq 3$. Our goal is to show that $\mathcal{G}$ has k edge-disjoint double S-subgraphs that extend $\mathcal{P}_k(v)$ of v such that every vertex in R is balanced-extendible. Let W be the set of neighbours of v in $V(\mathcal{G}) - S - R$ and B be the set of neighbours of v in $S \cup R$. By Lemma 2, each $w_i \in W$ is incident with exactly three edges and adjacent to exactly three vertices, so we let $N_{\mathcal{G}}(w_i) := \{v, x_i, y_i\}$ and call $\{x_i, y_i\}$ a **couple**. By Lemma 6, x_i and y_i are in $S \cup R$. For each $b_i \in B$, we denote by $c(b_i)$ the number of multiple edges between v and b_i.

Let G' be $\mathcal{G} - v - W$. Let Z be a minimum $(S \cup R - v)$-cut of G' and $\{C_1, \ldots, C_l\}$ be the connected components of $G' - Z$. We let $S_i := S \cap V(C_i)$, $R_i := R \cap V(C_i)$ and $B_i := B \cap V(C_i)$. Also, $c(B_i)$ denotes the sum of the $c(b)$ for $b \in B_i$ and X_i denotes the collection of couples with both vertices in C_i. By the minimality of Z, each edge e in Z connects two vertices in different components, and we call it a **crossing edge**. Similarly, a couple $\{x_i, y_i\}$ is a **crossing couple** if x_i and y_i are in different components, and we denote the collection of crossing couples by X_C.

Lemma 7. *$(S \cup R - v)$ is at most $(6k-1)$-edge-connected in G'.*

Proof. Since $|S \cup R| \geq 3$, $|S \cup R - v| \geq 2$. If $(S \cup R - v)$ is $6k$-edge-connected in G', then by Theorem 2, there are $2k$ edge-disjoint $(S \cup R - v)$-subgraphs $\{H_1'', \ldots, H_{2k}''\}$ in G'. Notice that since the union of two edge-disjoint $(S \cup R - v)$-subgraphs is a double $(S \cup R - v)$-subgraph, by setting $H_i' := H_{2i-1}'' \cup H_{2i}''$, $\{H_1', \ldots, H_k'\}$ are k edge-disjoint double $(S \cup R - v)$-subgraphs of G'. Now, let $H_i := H_i' \cup \{vb_j | vb_j \in E_i(v)\} \cup \{vw_j, w_jx_j | vw_j \in E_i(v)\}$. So, $E_i(v) \subseteq H_i$, and $H_i - v$ is a double $(S \cup R - v)$-subgraph that spans $N_{E_i}(v)$. Also, since $|E_i(v)| \geq 2$,

H_i is a double $(S \cup R)$-subgraph of $\mathcal{G}$. By Definition 1, $\{H_1, \ldots, H_k\}$ are k edge-disjoint double $(S \cup R)$-subgraphs of $\mathcal{G}$ that extend $\mathcal{P}_k(v)$, a contradiction. □

Lemma 8. *$G' - Z$ has 2 connected components.*

Proof. We just need to show that G' has at most 2 connected components, then the statement that $G' - Z$ has 2 connected components follows from the minimality of Z. Notice that from our construction of G' from $\mathcal{G}$, the set of neighbours of every vertex in $V(\mathcal{G}) - S - R$ that remained in G' is the same as in $\mathcal{G}$. Since $\mathcal{G}$ is connected, no component in G' contains only vertices in $V(\mathcal{G}) - S - R$. Therefore, it remains to show that there are at most two components in G' that contain vertices in $S \cup R$.

Suppose there are three connected components containing vertices in $S \cup R$. Let $u_1, u_2, u_3 \in S \cup R$ be vertices in C_1, C_2, C_3 respectively. Since u_1, u_2, u_3 have at least $(Q-2)k$ edge-disjoint paths to v and v is of degree Qk, there exists a vertex $w \in N_{\mathcal{G}}(v)$ such that u_1, u_2, u_3 all have a path to w in $\mathcal{G}$. If $w \in S \cup R$, then clearly u_1, u_2, u_3 are still connected in G', a contradiction. If $w \in V(\mathcal{G}) - S - R$, then w is of degree 3 by Lemma 2. Then there must exist a pair of vertices, say u_1 and u_2, both have a path to the same neighbour of w in G', a contradiction. □

5.3 Diverging Paths and Common Paths

Consider a vertex $u \neq v$ where $u \in S$. Since S is Qk-edge-connected in $\mathcal{G}$, by Menger's theorem, there are Qk edge-disjoint paths, denoted by $P(u) := \{P_1(u), \ldots, P_{Qk}(u)\}$, from u to v. Note that since v is of degree exactly Qk, each path in $P(u)$ uses exactly one edge in $E(v)$. Furthermore, since w_i is of degree 3 by Lemma 2, each w_i is used by exactly one path in $P(u)$. Similarly, for $u \neq v$ where $u \in R$. There are $(Q-2)k$ edge-disjoint paths from u to v denoted by $P(u) := \{P_1(u), \ldots, P_{(Q-2)k}(u)\}$. Again, we may assume that each path use exactly one edge in $E(v)$ and at most one vertex in W. Consider $P_i(u)$ induced in G', denoted by $P'_i(u)$. Let $P'(u) := \{P'_1(u), \ldots, P'_{Qk}(u)\}$ for $u \in S$, and similarly $P'(u) := \{P'_1(u), \ldots, P'_{(Q-2)k}(u)\}$ for $u \in R$. Notice that $P'(u)$ contains edge-disjoint paths in G', and we call them the **diverging paths** from u.

We plan to use the diverging paths from a and b for any two vertices $a, b \in S \cup R$ in the same component of $G' - Z$ to establish the connectivity of $S \cup R$ in each component of $G' - Z$. We say v_1 and v_2 have λ **common paths** if there are λ edge-disjoint paths starting from v_1, λ edge-disjoint paths starting from v_2, and an one-to-one mapping of the paths from v_1 to the paths from v_2 so that each pair of paths in the mapping ends in the same vertex. The following lemma gives a lower bound on the number of edge-disjoint paths between two vertices based on the number of their common paths.

Lemma 9. *[9] If v_1 and v_2 have $2\lambda + 1$ common paths in G, then there exist $\lambda + 1$ edge-disjoint paths from v_1 to v_2 in G.*

5.4 Both Components of $G' - Z$ Contain Vertices in S

In this subsection, we consider the case that both components contain some vertices in S. The lemmas in this section all share this assumption.

Lemma 10. *If both components of $G' - Z$ contain some vertices in S, then there are at least $Qk - 2|Z|$ crossing couples, that is, $|X_C| \geq Qk - 2|Z|$.*

Proof. Let $u_1 \in S$ be in C_1. In G', u_1 has at least $c(B_2)+|X_2|$ edge-disjoint paths in $P'(u_1)$ to C_2. Since Z is an edge-cut in G', it follows that $c(B_2) + |X_2| \leq |Z|$. Similarly, by considering a vertex $u_2 \in S$ in C_2, we have $c(B_1) + |X_1| \leq |Z|$. By Lemma 8, there are only two components in $G' - Z$. So $Qk = |X_C| + |X_1| + |X_2| + c(B_1) + c(B_2)$, and we have $|X_C| \geq Qk - 2|Z|$. □

Lemma 11. *If both components of $G' - Z$ contain some vertices in S, then $S_i \cup R_i$ is $(Q/2 - 14)k$-edge-connected in C_i of $G' - Z$.*

Proof. Consider any two vertices $a, b \in S_i \cup R_i$ in C_i. In G', $P'(a)$ has at least $|X_C| - 2k$ paths to different crossing couples. Among those paths, at most $|Z|$ of them may use edges in Z. So, in $G' - Z$, a has at least $|X_C| - 2k - |Z|$ edge-disjoint paths such that each starts at a and ends in different crossing couples, and similarly for b. Therefore, in $G' - Z$, a and b have at least $(|X_C| - 2k - |Z|) + (|X_C| - 2k - |Z|) - (|X_C|) = |X_C| - 4k - 2|Z| \geq Qk - 4k - 4|Z| \geq Qk - 28k$ common paths in C_i; the second last and the last inequality hold because of Lemma 10 and Lemma 7, respectively. By Lemma 9, a and b are $(Q/2 - 14)k$-edge-connected in C_i. □

Lemma 12. *If both components of $G' - Z$ contain some vertices in S, then $\mathcal{G}$ has k edge-disjoint double S-subgraphs $\{H_1, \ldots, H_k\}$ that extend $\mathcal{P}_k(v)$ such that every vertex in R is balanced-extendible.*

Proof. (Sketch) We pick arbitrarily $\min\{k, |Z|\}$ edges in Z and call them the *connecting edges*. For each connecting edge e with a endpoint $w \in V(\mathcal{G}) - S - R$ in C_i, we remove one edge e' in C_i which is incident with w (by Lemma 6, the other endpoint of e' must be black), and we call e' a *reserve edge* of e. Let the resulting component be C_i'. Since we remove at most k edges and $S_i \cup R_i$ is $(Q/2 - 14)k$-edge-connected in C_i by Lemma 11, each $S_i \cup R_i$ is $(Q/2 - 15)k$-edge-connected in C_i'. In particular, each $S_i \cup R_i$ is $6k$-edge-connected in C_i'. By Theorem 2, there are $2k$ edge-disjoint $(S_i \cup R_i)$-subgraphs in C_i'. So there are k edge-disjoint double $(S_i \cup R_i)$-subgraphs $\{H_1^i, \ldots, H_k^i\}$ in each C_i' for $i \in \{1, 2\}$.

Now we set $H_j := H_j^1 \cup H_j^2 \cup \{vb_i | vb_i \in E_j(v)\} \cup \{vw_i, w_ix_i, w_iy_i | vw_i \in E_j(v)\}$ for $1 \leq j \leq k$. Notice that $E_j(v) \subseteq E(H_j)$ and $H_j - v$ spans $N_{E_j}(v)$ for $1 \leq j \leq k$. Suppose there is a crossing couple $\{x_i, y_i\}$ such that $vw_i \in E_j(v)$, then H_j is also connected and thus is a $(S \cup R)$-subgraph of $\mathcal{G}$ that $E_j(v) \subseteq E(H_j)$ and $H_j - v$ is a $(S \cup R - v)$-subgraph that spans $N_{E_j}(v)$. Let's assume that $\{vw_1, \ldots, vw_{|X_C|}\}$ be the set of edges such that the corresponding couples are crossing. By Lemma 10, $|X_C| \geq Qk - 2|Z|$. Since $\mathcal{P}_k(v)$ is a balanced edge-subpartition, $|E_i(v)| \geq 2$ for $1 \leq i \leq k$. So, there are at most $\min\{k, |Z|\}$

classes of $\mathcal{P}_k(v)$ with no edges in $\{vw_1, \ldots, vw_{Qk-2|Z|}\}$. Hence there are at most $\min\{k, |Z|\}$ of H_j's, say $\{H_1, \ldots, H_{\min\{k,|Z|\}}\}$, are not connected by the crossing couples. Now, by adding each connecting edge and its reserve edge (if any) to a different H_j that has not been connected by a crossing couple, $\{H_1, \ldots, H_k\}$ are k edge-disjoint double $(S \cup R)$-subgraphs of $\mathcal{G}$ that extend $\mathcal{P}_k(v)$. (We skip the not-so-interesting case that $|S_i| = 1$ for some $i \in \{1, 2\}$, which needs to be handled separately.)

□

5.5 One Component Contains Only Vertices in R

Without loss of generality, we assume that C_1 contains vertices in S and C_2 contains only vertices in R. Lemma 13 and Lemma 14 are counterparts of Lemma 10 and Lemma 11, we omit the proofs here. However, it should be pointed out that we only have a weaker bound on the number of crossing couples in Lemma 13, and hence the strategy in Lemma 12 cannot be used. In Lemma 15, we use a different strategy to construct the desired subgraphs.

Lemma 13. *If C_2 contains only vertices in R, then there are at least $(Q-2)k-2|Z|$ crossing couples, that is, $|X_C| \geq (Q-2)k - 2|Z|$.*

Lemma 14. *If C_2 contains only vertices in R, then $S_1 \cup R_1$ is at least $(Q/2-15)k$-edge-connected in C_1 of $G' - Z$.*

Lemma 15. *If C_2 contains only vertices in R, then $\mathcal{G}$ has k edge-disjoint double S-subgraphs that extend $\mathcal{P}_k(v)$ such that every vertex in R is balanced-extendible.*

Proof. Let $E' := \{e_1, \ldots, e_{|X_2|+c(B_2)}\}$ be the set of edges incident to v in $\mathcal{G}$ so that either (i) the other endpoint of e_i is in B_2 or (ii) the other endpoint of e_i has both of its neighbour in C_2. Let $u_1 \in S$. From $P(u_1)$ in $\mathcal{G}$, there are $|X_2| + c(B_2)$ edge-disjoint paths from u_1 to v such that each uses exactly one edge in E'. From these paths, in $\mathcal{G} - C_1$, there are $|X_2| + c(B_2)$ edge-disjoint paths $P := \{P_1, \ldots, P_{|X_2|+c(B_2)}\}$ with the following property: each P_i starts from v and ends in some vertex of C_1, and $e_i \in P_i$. Since each vertex in $w \in V(C_1) - S_1 - R_1$ is of degree 3, by the minimality of $|Z|$, it has at most one neighbour in C_2. Therefore, each $w \in V(C_1)$ can be in at most one path in P. Now, for each $u_i \in V(C_1) - S_1 - R_1$, we remove one edge e' in C_1 which is incident with w and set $P'_i := P_i \cup \{e'\}$ (by Lemma 6, the other endpoint of e' must be in $S_1 \cup R_1$); otherwise, $P'_i := P_i$. So $P' := \{P'_1, \ldots, P'_{|X_2|+c(B_2)}\}$ are edge-disjoint paths with the following property: each P'_i starts from v and ends in some vertex $u_i \in S_1 \cup R_1$, and $e_i \in P'_i$. In constructing P' from P, we remove at most $|X_2| + c(B_2) \leq |Z| \leq 6k$ edges from C_1. Let the resulting component be C'_1. Since $S_1 \cup R_1$ is $(Q/2-15)k$-edge-connected in C_1 by Lemma 14, $S_1 \cup R_1$ is $(Q/2-21)k$-edge-connected in C'_1. In particular, C'_1 is $S_1 \cup R_1$ is $6k$-edge-connected in C'_1. By Theorem 2, there are $2k$ edge-disjoint $(S_1 \cup R_1)$-subgraphs in C'_1. So there are k edge-disjoint double $(S_1 \cup R_1)$-subgraphs $\{H^1_1, \ldots, H^1_k\}$ in C'_1. Now, we set $H_j := H^1_j \cup \{vb_i | vb_i \in E_j(v)\} \cup \{vw_i, w_ix_i, w_iy_i | vw_i \in E_j(v)\} \cup \{P_i | e_i \in E_j(v)\}$. Then, it is straightforward to check $\{H_1, \ldots, H_k\}$

are k edge-disjoint double S-subgraphs that extend $\mathcal{P}_k(v)$ so that every vertex in R is balanced-extendible. Indeed, every vertex in R_1 is balanced-extendible because it is of degree at least 2 in each H_i^1 and thus H_i; while every vertex in R_2 is balanced-extendible because it has degree at most $12k$ in H_j, as it is used by at most $6k$ paths from P'. This completes the proof of the lemma.

□

Putting Lemma 12 and Lemma 15 together shows that a minimal counterexample $\mathcal{G}$ of Theorem 5 does not exist, and this completes the proof of Theorem 5.

□

6 Concluding Remarks

As far as the algorithmic aspects go, it is straightforward to check that the proof yields a polynomial time constant factor approximation algorithm for the Steiner Forest Packing problem. Also, using the same technique as in [9], the approximation algorithm can be extended to the capacitated version of the Steiner Forest Packing, where each edge e has an capacity c_e so that at most c_e forests can use e (for the original problem, $c_e = 1$ for all $e \in E(G)$).

The following is a general problem that captures the Steiner Forest Packing problem. Given an undirected multigraph G and a connectivity requirement r_{uv} for each pair of vertices $u, v \in V(G)$, find a largest collection of edge-disjoint subgraphs of G such that in each subgraph there are r_{uv} edge-disjoint paths from u to v for all $u, v \in V(G)$. Since connectivity is transitive, it is not difficult to see that (see [1]) Theorem 1 is equivalent to the following:

Theorem 6. *Given an undirected multigraph G and a connectivity requirement $r_{uv} \in \{0, 1\}$ for $u, v \in V(G)$. If there are $Qk \cdot r_{uv}$ edge-disjoint paths for all $u, v \in V(G)$, then there are k edge-disjoint forests such that in each forest there is r_{uv} path between u, v for all $u, v \in V(G)$.*

I conjecture that Theorem 6 can be generalized to arbitrary non-negative integer connectivity requirements:

Conjecture 1. Given an undirected multigraph G and a connectivity requirement r_{uv} for each pair of vertices $u, v \in V(G)$. There exists a universal constant c so that the following holds. If there are $ck \cdot r_{u,v}$ edge-disjoint paths for all $u, v \in V(G)$, then there are k edge-disjoint subgraphs $H_1, \ldots, H_k$ in G such that in each subgraph there are r_{uv} edge-disjoint paths between u and v for all $u, v \in V(G)$.

It would be interesting to first verify Conjecture 1 for Eulerian graphs (with $c = 2$?). This may also yield insights into the generalized Steiner network problem, for which the only constant factor approximation algorithm is due to Jain's iterative rounding technique [5].

Acknowledgment

Part of this work was done when the author visited the Operation Research Department of the Eötvös Loránd University; the author is very grateful to András Frank, Tamás Király and Jácint Szabó for their hospitality, and the Egerváry Research Group (EGRES) for providing an excellent research environment. The author would also like to thank Mike Molloy for helpful comments and Volker Kaibel for his help in preparing the final version of this paper.

References

1. C. Chekuri, B. Shepherd. Approximate integer decompositions for undirected network design problems. *Manuscript*, 2004.
2. J. Cheriyan, M. Salavatipour. Hardness and approximation results for packing Steiner trees problems. *Proceedings of the 12th Annual European Symposium on Algorithms (ESA)*, LNCS Vol 3221, pp.180-191, 2004.
3. A. Fránk, T. Király, M. Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discrete Applied Mathematics*, 131, pp.373-383, 2003.
4. M. Goemans, D. Williamson. A general approximation technique for constrained forests problems. *SIAM Journal on Computing*, 24, pp.296-317, 1995.
5. K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21, pp.39-60, 2001.
6. K. Jain, M. Mahdian, M.R. Salavatipour. Packing Steiner trees. *Proceedings of the 14th Annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pp.266-274, 2003.
7. M. Kriesell. Edge-disjoint trees containing some given vertices in a graph. *J. Combin. Theory, Series B*, 88, pp.53-63, 2003.
8. M. Kriesell. Disjoint Steiner trees in graphs without large bridges. *manuscript*, 2004.
9. L.C. Lau. An approximate max-Steiner-tree-packing min-Steiner-cut theorem. *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 61-70, 2004.
10. W. Mader. A reduction method for edge-connectivity in graphs. *Ann. Discrete Math.*, 3, pp.145-164, 1978.
11. K. Menger. Zur allgemeinen Kurventheorie. *Fund. Math.*, 10, pp.95-115, 1927.
12. C.St.J.A. Nash-Williams. Edge disjoint spanning trees of finite graphs. *J. London Math. Soc.*, 36, pp.445-450, 1961.
13. L. Petingi, J. Rodriguez. Bounds on the maximum number of edge-disjoint Steiner trees of a graph. *Congressus Numerantium*, 145, pp.43-52, 2000.
14. W.T. Tutte. On the problem of decomposing a graph into connected factors. *J. London Math. Soc.*, 36, pp.221-230, 1961.

Bidimensional Packing by Bilinear Programming

Alberto Caprara[1], Marco Locatelli[2], and Michele Monaci[1]

[1] DEIS, Università di Bologna,
Viale Risorgimento 2, 40136 Bologna, Italy
{acaprara, mmonaci}@deis.unibo.it
[2] Dipartimento di Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino, Italy
locatelli@di.unito.it

Abstract. We consider geometric problems in which rectangles have to be packed in (identical) squares, that turn out to be very hard in practice and for which ILP formulations in which variables specify the coordinates in the packing perform very poorly. While most methods developed until the end of last century are based on simple geometric considerations, a recent landmark result of Fekete and Schepers suggests to put these geometric aspects aside and use the most advanced tools for the 1-dimensional case. In this paper we make additional progress in this direction, especially on the basic question "Does a given set of rectangles fit in a square?", that turns out to be the bottleneck of all the approaches known.

Given a set of rectangles and the associated convex hull of the incidence vectors of rectangle subsets that fit in a square, we derive a wide class of valid inequalities for this convex hull from a complete description of the two knapsack polytopes associated with the widths and the heights of the rectangles, respectively. Additionally, we illustrate how to solve the associated separation problem as a bilinear program, for which we develop a solution method that turns out to be fast in practice, and show that integer solutions that satisfy all these inequalities generally correspond to vertices of the original convex hull. The same tools are used to derive lower bounds for the 2-dimensional bin packing problem, corresponding to the determination of an optimal pair of so-called dual feasible functions, that in many cases equal the lower bounds obtained by the customary set covering formulation (for which column generation is very hard) being computable within times that are orders of magnitude smaller.

All our results extend immediately to the general problem of packing d-dimensional parallelepipeds in hypercubes.

1 Introduction

Bidimensional packing problems, that play an important role in practical cutting and packing applications, have been studied in the literature since the early 60s. These problems share the well-known characteristic that they are extremely complicated to model from a mathematical viewpoint unless some strong constraints

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 377–391, 2005.

on the form of the packings produced are imposed, such as two- or three-stage patterns. In particular, the natural ILP models in which one specifies the coordinates of each item in the packing tend to be of very large size and to have very weak LP relaxations. Due to this inherent complication, mainly heuristic algorithms were proposed for most of these problems, some of which were analyzed from a worst-case point of view. Only starting from the late 90s serious attempts were made to solve some of these problems to proven optimality, using rather simple combinatorial/geometric considerations to derive bounds on the optimum rather than using any ILP. This led to some achievement, but there seems to be wide space for improvement.

The hardest problems in this context arise when items have to be packed in finite bins, the main representatives requiring to pack a given set of rectangles in squares. In particular, the 2-dimensional bin packing problem requires to pack all the given rectangles in the minimum number of identical squares, and the 2-dimensional knapsack problem to pack a maximum-profit subset of the given rectangles in a single square. A recent breakthrough in the study of these problems [11, 24] was the observation that the so-called dual feasible functions, corresponding to dual solutions of the customary LP relaxation of 1-dimensional bin packing, can also be used to compute bounds for 2-dimensional packing. So far, however, the choice of the dual feasible functions to use in these cases was limited to a (small) number of predefined ones. Our paper builds on the idea of finding the best dual-feasible functions for the instance at hand.

The state-of-the-art exact approaches to 2-dimensional bin packing are the purely combinatorial branch-and-bound approaches by [20, 19] and the branch-and-price approach of [23]. For 2-dimensional knapsack, the best exact approaches are the branch-and-bound approaches in [12, 8], both using fairly simple bounds. For all of these methods with the exception of [23], branching conditions do not specify the coordinates of the rectangles in a packing but simply a set of rectangles that one intends to pack in a square, and then one has to test whether this set actually fits in a square by a separate enumeration. At present this feasibility test is the most time consuming part of the algorithms, the most promising approach to it being in our opinion based on the graph-theoretic characterization of feasible packings by [10], that does not have to specify the coordinates of the rectangles and can be used to drive enumeration [12].

1.1 Our Contribution

According to the above discussion, any progress towards the answer to the following question: "Does a given set of rectangles fit in a square?" is expected to lead to progress towards the solution of the two hard problems mentioned above. In this paper, we face the problem from a polyhedral viewpoint, considering, for a given set of rectangles, the 2-dimensional knapsack polytope, defined as the convex hull of the incidence vectors of rectangle subsets that fit in a square. Building on the results in [11, 24], the main results presented in this paper, all of which extend immediately to the general problem of packing d-dimensional parallelepipeds in hypercubes, are the following:

- We show that, given a set of rectangles and two inequalities $\sum_j \alpha_j x_j \leq 1$ and $\sum_j \beta_j x_j \leq 1$ valid for the (binary) knapsack polytopes associated, respectively, with the widths and the heights of the rectangles, the inequality

$$\sum_j \alpha_j \beta_j x_j \leq 1 \tag{1}$$

  is valid for the 2-dimensional (binary) knapsack polytope associated with the rectangles. This also implies that the value

$$\sum_j \alpha_j \beta_j \tag{2}$$

  is a valid lower bound on the optimal value of the 2-dimensional bin packing associated with the rectangles.
- We show that separation of inequalities (1), as well as the computation of α, β leading to the highest lower bound (2), can be stated as a bilinear programming problem, that we solve by a canonical LP-based approach that turns out to be fast in practice.
- For 2-dimensional (binary) knapsack, we show that for most instances of the literature the optimal value coincides with the upper bound given by the optimum over the convex hull of the binary vectors that satisfy all inequalities (1).
- For 2-dimensional bin packing, we show that for several instances of the literature the best lower bound (2) dominates all combinatorial bounds (being computable in comparable time) and equals the (optimum and the) lower bound obtained by the naive generalization of the customary LP relaxation for the 1-dimensional case, which is generally order of magnitudes slower to solve.
- Noting that the first of the above points was essentially stated in terms of valid inequalities for the *unbounded* knapsack polytope in [11, 24], we show both that the problem in the second point is easier in this case, but also that the bounds in the third and fourth points are weaker for several instances of the literature.

The points above are not presented in this order in the paper, namely we will address first the 2-dimensional bin packing case and then the 2-dimensional knapsack case. For space reasons, several proofs are deferred to the full paper.

1.2 The Problems Considered

Given an optimization problem P and an instance I of P, we will let $\mathrm{opt}_{\mathrm{P}}(I)$ denote the optimal value of P for instance I.

The 1-dimensional problems that we will consider throughout the paper work with a set of *items* of a specified *size* to be packed in *bins*, guaranteeing that the total size of the items packed in each bin does not exceed its size.

Definition 1. *We will denote by $\mathcal{I}$ the collection of all item sets $I = (n, s)$, where $n \in \mathbb{N}$ is the number of items and, for $j = 1, \ldots, n$, $s_j \in (0, 1]$ is the size of item j.*

Definition 2. *The* Bin Packing Problem (BP)*, given $I \in \mathcal{I}$ and infinitely many bins of size* 1*, calls for packing all the items in I in the minimum number of bins.*

Definition 3. *The* (Binary) Knapsack Problem (KP)*, given $I \in \mathcal{I}$, along with a profit $p_j > 0$ for the jth item in I, and a single bin (knapsack) of size* 1*, calls for packing a subset of the items in I with maximum profit in the bin. The* Unbounded KP (UKP) *is the variant of KP in which each item is available in unlimited number of copies, and one can pack any number of copies of each item in the bin.*

Definition 4. *Given $I = (n, s) \in \mathcal{I}$, a* bounded feasible pattern *is a vector $b \in \{0,1\}^n$ such that $\sum_{j=1}^{n} b_j s_j \leq 1$, i.e. the incidence vector of a feasible solution of KP associated with I. We will denote by $\mathcal{B}(I)$ the collection of all binary feasible patterns associated with I, and by $P_{\mathrm{KP}}(I) := \mathrm{conv}(\mathcal{B}(I))$ the corresponding KP polytope.*

Definition 5. *Given $I = (n, s) \in \mathcal{I}$, an* unbounded feasible pattern *is a vector $u \in \mathbb{Z}_+^n$ such that $\sum_{j=1}^{n} u_j s_j \leq 1$, i.e. the incidence vector of a feasible solution of UKP associated with I. We will denote by $\mathcal{U}(I)$ the collection of all unbounded feasible patterns associated with I, and by $P_{\mathrm{UKP}}(I) := \mathrm{conv}(\mathcal{U}(I))$ the corresponding UKP polytope.*

The results of this paper concern the two dimensional generalizations of the problems above, all of which work with a set of *rectangles* of specified *width* and *height* to be packed in *squares*. The packing in each square must be *orthogonal without rotation*, i.e. the rectangles packed must not overlap and all the widths of the rectangles packed must be parallel to the same edge of the square.

Definition 6. *We will denote by $\mathcal{R}$ the collection of all rectangle sets $R = (n, w, h)$, where $n \in \mathbb{N}$ is the number of rectangles and, for $j = 1, \ldots, n$, $w_j, h_j \in (0, 1]$ are the width and the height of rectangle j.*

Definition 7. *The* 2-Dimensional BP (2BP), 2-Dimensional KP (2KP) *and* 2-Dimensional UKP (2UKP) *are the generalizations of BP, KP and UKP, respectively, in which one is given a set $R \in \mathcal{R}$ (along with a profit $p_j > 0$ for the jth rectangle in R in the last two cases) instead of a set $I \in \mathcal{I}$, and bins that are* squares *of size* 1*.*

Definition 8. *Given $R = (n, w, h) \in \mathcal{R}$, we will denote by $P_{\mathrm{2KP}}(R)$ and by $P_{\mathrm{2UKP}}(R)$ the convex hulls of the incidence vectors of the feasible solutions of 2KP and 2UKP, respectively.*

1.3 Bilinear Programming

Definition 9. *A* (Disjoint) Bilinear Program (BLP)*, given $c \in \mathbb{R}^{n_1}$, $d \in \mathbb{R}^{n_2}$, $E \in \mathbb{R}^{n_1 \times n_2}$, $A \in \mathbb{R}^{m_1 \times n_1}$, $B \in \mathbb{R}^{m_2 \times n_2}$, $a \in \mathbb{R}^{m_1}$, $b \in \mathbb{R}^{m_2}$, calls for $x \in \mathbb{R}^{n_1}$ and $y \in \mathbb{R}^{n_2}$ that solve:*

$$\max\{c^T x + d^T y + x^T E y : Ax \leq a, By \leq b\}. \tag{3}$$

Informally, a BLP is the generalization of LP in which there are two variable vectors x and y, distinct linear constraints are imposed separately on x and y, and the objective function involves products of components of x and y. It is known that the problem is strongly NP-hard in general. Solution approaches to the problem have been widely studied in the literature and are surveyed in [2, 14]. Among these, we outline a simple branch-and-bound approach [21] that will be convenient for our purposes. We restrict attention to the case in which the upper and lower bounds on x and y implied by the linear constraints are finite:

$$l^x \leq x \leq u^x, \quad l^y \leq y \leq u^y, \tag{4}$$

and $E \geq 0$, i.e. all coefficients for quadratic terms in the objective function are nonnegative. Consider the linearization of the objective function obtained by replacing each term $e_{ij}x_iy_j$ by $e_{ij}z_{ij}$, where z_{ij} are new variables that are bounded by the following additional linear constraints:

$$z_{ij} \leq u_j^y x_i + l_i^x y_j - l_i^x u_j^y, \quad z_{ij} \leq u_i^x y_j + l_j^y x_i - l_j^y u_i^x, \quad i = 1, \ldots, n_1, j = 1, \ldots, n_2, \tag{5}$$

that are easily seen to be implied by $z_{ij} = x_iy_j$. Actually, all linear constraints of the form $z_{ij} \leq \alpha x_i + \beta y_j + \gamma$ that are implied by (4) and $z_{ij} \leq x_iy_j$ are dominated by (5). Moreover, if either x_i or y_j attains its lower or upper bound, (5) are equivalent to $z_{ij} \leq x_iy_j$.

The branch-and-bound algorithm proceeds by first solving the the LP relaxation obtained by the above substitution, i.e.

$$\max\{c^T x + d^T y + \textstyle\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} e_{ij} z_{ij} : (5), Ax \leq a, By \leq b\}. \tag{6}$$

If in the optimal solution (x^*, y^*, z^*) of (6) each of the x and y variables attains either its lower or upper bound, the solution is optimal for (3). Otherwise, one considers an x or y variable whose optimal value is strictly between its bounds and branches by changing the bounds on that variable, noting that the bounds affect constraints (5) and therefore the objective function value. We refer to the references above and to the full paper for further details.

2 Fractional Bin Packing, Dual Feasible Functions, and Knapsack Separation

There are two commonly used versions of the customary LP relaxation of BP, which are often confused with each other. Since using one version or the other often makes a lot of difference in our application, and in fact it is sometimes better to use both, we point out the differences in this section in order to avoid confusion later. In the classical version [15], the LP has one variable x_u associated with each unbounded feasible pattern $u \in \mathcal{U}(I)$:

$$\min\{\textstyle\sum_{u \in \mathcal{U}(I)} x_u : \sum_{u \in \mathcal{U}(I)} u_j x_u \geq 1 \ (j = 1, \ldots, n), \quad x \geq 0\}, \tag{7}$$

whose dual is:

$$\max\{\textstyle\sum_{j=1}^{n} \pi_j : \sum_{j=1}^{n} u_j \pi_j \leq 1 \ (u \in \mathcal{U}(I)), \quad \pi \geq 0\}. \tag{8}$$

Definition 10. *The* Unbounded Fractional BP (UFBP) *is defined by LP (7).*

Definition 11. *The* (Bounded) Fractional BP (FBP) *is the variant of UFBP in which bounded feasible patterns in $\mathcal{B}(I)$ rather than unbounded feasible patterns in $\mathcal{U}(I)$ are considered.*

Fact 1. *Given $I \in \mathcal{I}$, $\mathrm{opt}_{\mathrm{BP}}(I) \geq \mathrm{opt}_{\mathrm{FBP}}(I) \geq \mathrm{opt}_{\mathrm{UFBP}}(I)$.*

Concerning the complexity of UFBP and FBP, the following is a well known result:

Proposition 1. *Both UFBP and FBP are weakly NP-hard and can be solved in time pseudopolynomial in the size of I.*

This follows from the well-known equivalence between separation and optimization [16] and the easy observation that the column generation problems (or, equivalently, the separation problems for the dual) correspond to the recognition version of UKP and KP, respectively, as already noted in [15]. The lower bound given by the optimal solution value of UFBP (and therefore also of FBP) turns out to be extremely strong in practice, as partially explained by the following theoretical result, which is implicit in [13, 18] and made explicit in [7]:

Theorem 1. $\lim_{z \to \infty} \sup_{I \in \mathcal{I}: \mathrm{opt}_{\mathrm{BP}}(I) \geq z} \dfrac{\mathrm{opt}_{\mathrm{BP}}(I)}{\mathrm{opt}_{\mathrm{UFBP}}(I)} = 1.$

Dual solutions of UFBP and FBP can have different interpretations. The following notion is introduced in [9]:

Definition 12. *A* Dual Feasible Function *(DFF), is a function $f : [0, 1] \to [0, 1]$ such that $\sum_{i=1}^{m} f(x_i) \leq 1$ for each $m \in \mathbb{N}$ and $x \in \mathbb{R}_+^m$ such that $\sum_{i=1}^{m} x_i \leq 1$. We let $\mathcal{D}$ denote the collection of all DFFs.*

DFFs can be used to compute lower bounds for BP, since clearly

Fact 2. *Given $I = (n, s) \in \mathcal{I}$, $\mathrm{opt}_{\mathrm{BP}}(I) \geq \sum_{j=1}^{n} f(s_j)$ for each $f(\cdot) \in \mathcal{D}$.*

The approach proposed by [9] and then followed by other authors is to try a few DFFs and take the best lower bound obtained. From a different perspective, one may try to solve the following problem:

Definition 13. *The* DFF Problem (DP)*, given $I = (n, s) \in \mathcal{I}$, calls for $f(\cdot) \in \mathcal{D}$ that solves: $\sup_{f(\cdot) \in \mathcal{D}} \sum_{j=1}^{n} f(s_j)$.*

The following easy observation is pointed out in [9] (see [7] for a formal proof).

Lemma 1. *Consider $I = (n, s) \in \mathcal{I}$, assuming $s_1 \leq s_2 \leq \ldots \leq s_n$. For every $f(\cdot) \in \mathcal{D}$ the vector $(f(s_1), \ldots, f(s_n))$ is a feasible dual solution of UFBP.*

Conversely, for every feasible dual solution π of UFBP, the function $f(\cdot)$ defined by $f(x) := \pi_j$ for $x \in [s_j, s_{j+1})$ and $j = 0, \ldots, n$ (letting $s_0 := 0$ and $s_{n+1} := 1$) belongs to $\mathcal{D}$.

This implies:

Proposition 2. *UFBP and DP are equivalent.*

There is another fundamental interpretation of UFBP and FBP, namely their feasible solutions correspond to valid inequalities for P_{UKP} and P_{KP}, respectively. We state the formal results for FBP and P_{KP}, noting that they have immediate counterparts for UFBP and P_{UKP}.

Fact 3. *Given $I = (n, s) \in \mathcal{I}$, $\pi \in \mathbb{R}^n_+$ is a feasible solution of the dual of FBP if and only if inequality $\sum_{j=1}^n \pi_j x_j \leq 1$ is valid for $P_{\mathrm{KP}}(I)$.*

The corresponding separation problem is the variant of FBP in which we allow the objective function coefficients to be arbitrary nonnegative values, which does not change the solution approach outlined above.

Fact 4. *Given $I = (n, s) \in \mathcal{I}$ and $x^* \in [0, 1]^n$, the optimal solution of the dual of FBP with the modified objective function $\max \sum_{j=1}^n x_j^* \pi_j$ is at most 1 if and only if $x^* \in P_{\mathrm{KP}}(I)$. If this is not the case, the optimal dual FBP solution π^* defines the valid inequality $\sum_{j=1}^n \pi_j^* x_j \leq 1$ for $P_{\mathrm{KP}}(I)$ which is most violated by x^*.*

3 From 2-Dimensional Bin Packing ...

As anticipated, while in the 1-dimensional case one generally starts with KP and then moves to BP, we start from 2BP in the 2-dimensional case. For this case, the straightforward generalizations of UFBP and FBP are the following.

Definition 14. *The* 2-Dimensional Unbounded FBP (2UFBP) *and* 2-Dimensional FBP (2FBP) *are the generalizations of FBP and UFBP in which one is given a set $R \in \mathcal{R}$ instead of a set $I \in \mathcal{I}$, and bins that are* squares *of size* 1.

In particular, 2UFBP and 2FBP contain one variable for each feasible solution of 2UKP and 2KP, respectively. The clear disadvantage of solving the corresponding LPs is that the column generation problems correspond to a 2UKP and a 2KP, that at present are very hard in practice. So far there were many attempts to use one of these generalizations anyway, but the only reference reporting successful results is [23].

3.1 2-Dimensional Bin Packing and Dual Feasible Functions

Rather than trying to work on the obvious extensions above, we will see how DFFs can be extended here. While DFFs in the BP case are just a different way (although useful in many respects) to see the dual LP relaxation (8), they are a basic tool to tackle 2BP. The milestone result in this direction is from [11], for which a simple direct proof is given in [24] (and that holds for any dimension):

Theorem 2. *Given $R = (n, w, h) \in \mathcal{R}$ such that the corresponding rectangles can be packed in a square of size* 1, $\sum_{j=1}^{n} f(w_j)g(h_j) \leq 1$ *for every* $f(\cdot), g(\cdot) \in \mathcal{D}$.

This has the following immediate corollary:

Corollary 1. *Given* $R = (n, w, h) \in \mathcal{R}$, $\text{opt}_{\text{2BP}}(R) \geq \sum_{j=1}^{n} f(w_j)g(h_j)$ *for every* $f(\cdot), g(\cdot) \in \mathcal{D}$.

Following the approach to BP in [9], the proposal in [11] is to try a number of possibilities for $f(\cdot)$ and $g(\cdot)$ and choose the pair leading to the best lower bound on the 2BP solution value. The purpose of this section is show how to find the best pair *at all*.

Definition 15. *The* 2-Dimensional DFF Problem (2DP)*, given* $R = (n, w, h) \in \mathcal{R}$*, calls for* $f(\cdot), g(\cdot) \in \mathcal{D}$ *that solve:* $\sup_{f(\cdot),g(\cdot)\in\mathcal{D}} \sum_{j=1}^{n} f(w_j)g(h_j)$.

The following easy observation shows that the bound that we can hope for is not going to be better than the one associated with 2UFBP:

Fact 5. *Given* $R \in \mathcal{R}$, $\text{opt}_{\text{2BP}}(R) \geq \text{opt}_{\text{2UFBP}}(R) \geq \text{opt}_{\text{2DP}}(R)$.

On the other hand, the key issue is that 2DP turns out to be much easier to solve than 2UFBP.

Concerning the worst-case ratio between the optimal 2BP and 2DP values, the following holds, where $\zeta \approx 1.69$ is the worst-case ratio of the famous *harmonic algorithm* for BP [17]:

Proposition 3. $\lim_{z\to\infty} \sup_{R\in\mathcal{R}:\text{opt}_{\text{2BP}}(R)\geq z} \dfrac{\text{opt}_{\text{2BP}}(R)}{\text{opt}_{\text{2DP}}(R)} \in [9/8, \zeta]$.

3.2 The 2-Dimensional DFF Problem as a BLP

Recalling the correspondence between DFFs and solutions of UFBP pointed out by Lemma 1, we can formulate 2DP as follows. Given $R = (n, w, h) \in \mathcal{R}$ we consider the (1-dimensional) item sets $I^w := (n, w) \in \mathcal{I}$ and $I^h := (n, h) \in \mathcal{I}$ with corresponding unbounded feasible pattern collections $\mathcal{U}(I^w)$ and $\mathcal{U}(I^h)$. A BLP formulation of 2DP is:

$$\begin{aligned} \max & \sum_{j=1}^{n} \pi_j \rho_j, \\ & \sum_{j=1}^{n} u_j \pi_j \leq 1, \quad u \in \mathcal{U}(I^w), \\ & \sum_{j=1}^{n} u_j \rho_j \leq 1, \quad u \in \mathcal{U}(I^h), \\ & \pi_j, \rho_j \geq 0, \quad j = 1, \ldots, n. \end{aligned} \tag{9}$$

It is immediate to see that 2DP is weakly NP-hard, as a corollary of Proposition 1, and we conjecture that it is strongly NP-hard. Our solution approach to 2DP follows the branch-and-bound scheme outlined in Section 1.3.

Definition 16. *We will denote by 2LDP the LP relaxation (6) associated with BLP (9).*

Of course, as both the constraints on the π and ρ variables are exponentially many, we adopt a cutting plane approach to solve 2LDP, the separation problem being clearly an UKP. Another obvious corollary of Proposition 1 is:

Proposition 4. *2LDP is weakly NP-hard and can be solved in time pseudopolynomial in the size of R.*

A key issue for the strength of the LP relaxation is the tightness of the bounds on the variables. Rather than setting all lower bounds to 0 and all upper bounds to 1, we can use:

Fact 6. *Given $R = (n, w, h) \in \mathcal{R}$, the following are valid lower and upper bounds on the optimal 2DP variables:*

$$\begin{cases} u_j^\pi = 1/k, & \text{if } w_j \in (1/(k+1), 1/k] \text{ with } k \in \mathbb{N}, k \geq 2, \\ l_j^\pi = 1 - 1/k, & \text{if } w_j \in [1 - 1/k, 1 - 1/(k+1)) \text{ with } k \in \mathbb{N}, k \geq 2. \end{cases} \tag{10}$$

as well as their counterparts for u_j^ρ and l_j^ρ.

With the above lower and upper bounds, we can show that the ratio between the optimal values of 2DP and 2LDP is bounded by a constant, where the definition of $\eta \approx 1.35$ is given in the full paper.

Proposition 5. $\sup_{R \in \mathcal{R}} \dfrac{\mathrm{opt}_{\mathrm{2LDP}}(R)}{\mathrm{opt}_{\mathrm{2DP}}(R)} \in [\eta, 2]$.

3.3 The Bounded Version

Recalling that FBP is a stronger version of UFBP, it is natural to define the following BLP:

$$\begin{aligned} \max & \sum_{j=1}^{n} \pi_j \rho_j, \\ & \sum_{j=1}^{n} u_j \pi_j \leq 1, \quad u \in \mathcal{B}(I^w), \\ & \sum_{j=1}^{n} u_j \rho_j \leq 1, \quad u \in \mathcal{B}(I^h), \\ & \pi_j, \rho_j \geq 0, \quad j = 1, \ldots, n. \end{aligned} \tag{11}$$

The fact that also the optimal solution of this BLP yields a valid lower bound for 2BP is due to the following variant (in fact, generalization) of Theorem 2:

Theorem 3. *Given $R = (n, w, h) \in \mathcal{R}$ and any subset $S \subseteq \{1, \ldots, n\}$ such that the corresponding rectangles can be packed in a square of size 1, $\sum_{i \in S} \pi_i \rho_i \leq 1$ for every feasible solution (π, ρ) of (11) associated with R.*

We therefore define the following strengthening of 2DP, whose optimal solution yields a better lower bound for 2BP:

Definition 17. *The* 2-Dimensional Strengthened DFF Problem (2SDP)*, given $R = (n, w, h) \in \mathcal{R}$, is defined by (11).*

The following is the counterpart of Fact 5:

Fact 7. *Given $R \in \mathcal{R}$, $\text{opt}_{\text{2BP}}(R) \geq \text{opt}_{\text{2FBP}}(R) \geq \text{opt}_{\text{2SDP}}(R) \geq \text{opt}_{\text{2DP}}(R)$.*

Our solution approach to 2SDP is perfectly analogous to the one for 2DP, with the difference that Facts 6 does not hold any more, leading to weaker LP relaxations and making 2SDP generally more difficult to solve than 2DP.

3.4 Computational Results

We implemented the above methods in *C* using CPLEX 8.1 as LP solver. In order to check the appropriateness of our approach, we first verified that our method widely outperforms a state-of-the-art code for general BLP by [3, 1].

As benchmark, we considered the 500 2BP instances from the literature [4, 20]. For each instance R in this set, we considered the best lower bound known L^* [5, 6], including the lower bounds proposed in [11] but *excluding* the one obtained by solving 2FBP as in [23], and the best known heuristic solution value U^* [22]. In order to extract a representative set, we considered the 62 instances for which (i) $L^* < U^*$ (actually $L^* = U^* - 1$ in all cases) and (ii) $\text{opt}_{\text{2FBP}}(R) = U^*$, i.e. those for which optimality was proved by solving 2FBP. All these instances are given in Table 1, where we report the (rounded-up) values of 2DP, 2SDP and 2FBP along with the corresponding running times, in seconds on a Pentium M 1600 MHz, where for 2FBP we used our own implementation since we do not have the results of [23] for single instances (anyway, looking at the averages, the performances of our implementation appear to be similar to those in [23]).

The table shows that the 2DP running time, on average equal to 0.20, is notably smaller than the others and consistently small for all instances. In 17 cases, the associated 2BP lower bound is sufficient to prove optimality without resorting to the other two methods. The running time of 2SDP is higher, on average equal to 3.95, and less stable, but reasonably small in all cases. In 12 cases for which 2DP failed to prove optimality, 2SDP made it. For the remaining 33 cases optimality is proved by 2FBP, whose running time is much larger, on average equal to 1898.26, and very much dependent on the instance. In summary, for about half of the instances that were open before solving 2FBP, optimality can be proved within a time that is orders of magnitude smaller by running 2DP and then possibly 2SDP. There are 22 instances R out of the 500 for which $\text{opt}_{\text{2FBP}}(R) < U^*$ (actually $L^* = \text{opt}_{\text{2FBP}}(R) = U^* - 1$) that, as far as we know, remain open.

Table 1. 2BP, instances from the literature

R	n	L^*	U^*	2DP		2SDP		2FBP	
				Value	Time	Value	Time	Value	Time
1_40_1	40	9	10	10	0.00	10	0.04	10	3.68
1_40_2	40	11	12	11	0.10	12	0.24	12	0.02
1_40_3	40	16	17	16	0.00	16	0.12	17	0.00
1_60_1	60	22	23	23	0.00	23	0.12	23	0.03
1_60_2	60	18	19	19	0.01	19	0.48	19	0.46
1_60_7	60	15	16	16	0.00	16	0.60	16	0.67
1_80_1	80	24	25	25	0.01	25	0.75	25	0.04
3_20_3	20	5	6	5	0.01	5	0.08	6	0.04
3_20_6	20	6	7	6	0.00	7	0.04	7	0.01
3_40_10	40	7	8	7	0.03	7	0.52	8	1.33
3_60_2	60	12	13	13	0.08	13	1.52	13	13.53
3_60_3	60	13	14	13	0.09	13	1.84	14	1.49
3_60_4	60	14	15	15	0.02	15	1.45	15	0.33
3_80_3	80	17	18	18	0.03	18	3.35	18	1.32
3_80_5	80	16	17	16	0.25	16	4.09	17	22.51
3_100_3	100	18	19	19	0.42	19	9.57	19	440.72
3_100_9	100	21	22	22	0.08	22	7.18	22	0.67
5_40_4	40	12	13	12	0.02	12	0.54	13	1.54
5_40_5	40	13	14	13	0.10	13	0.61	14	0.55
5_40_9	40	9	10	9	0.07	9	0.69	10	3.28
5_60_6	60	15	16	15	0.22	15	2.07	16	10.86
5_60_10	60	23	24	23	0.04	23	1.49	24	0.57
5_80_2	80	22	23	23	0.12	23	5.33	23	112.63
5_80_3	80	24	25	24	0.10	24	4.79	25	2.44
5_80_5	80	22	23	22	0.28	22	4.75	23	1.92
5_80_6	80	25	26	25	0.09	25	4.57	26	1.62
5_80_9	80	26	27	26	0.17	26	3.45	27	9.27
5_80_10	80	22	23	23	0.32	23	5.12	23	10.41
5_100_1	100	23	24	24	0.46	24	10.20	24	331.23
5_100_2	100	28	29	28	0.33	28	10.41	29	66.92
7_20_6	20	5	6	5	0.01	6	0.09	6	0.30
7_40_3	40	9	10	9	0.13	9	0.58	10	5.05
7_40_7	40	11	12	11	0.03	12	0.39	12	4.33
7_60_5	60	14	15	14	0.06	15	0.79	15	17.03
7_80_1	80	20	21	20	0.30	20	3.74	21	637.01
7_80_3	80	20	21	20	0.10	20	5.09	21	103.52
7_80_5	80	23	24	23	0.07	24	1.98	24	3.54
7_80_6	80	22	23	22	0.09	23	1.93	23	226.20
7_80_8	80	22	23	22	0.07	23	2.61	23	15.91
7_80_9	80	23	24	23	0.09	23	5.20	24	75.87
7_100_3	100	24	25	24	0.08	25	4.07	25	18.76
7_100_4	100	26	27	26	0.10	27	3.69	27	13.95
8_40_1	40	11	12	11	0.04	11	0.46	12	1.41
8_60_9	60	16	17	16	0.07	17	1.19	17	2.85
8_80_3	80	20	21	20	0.07	20	4.85	21	2.47
8_100_1	100	26	27	26	0.10	26	11.40	27	13.64
8_100_4	100	30	31	30	0.11	30	10.03	31	44.02
8_100_6	100	26	27	26	0.11	26	7.60	27	3.87
8_100_7	100	25	26	25	0.69	25	8.67	26	37.76
10_20_4	20	4	5	4	0.02	5	0.06	5	0.09
10_20_10	20	2	3	3	0.08	3	0.10	3	4545.67
10_40_2	40	7	8	7	0.14	7	0.87	8	20.26
10_40_6	40	5	6	6	0.07	6	0.90	6	3070.52
10_60_4	60	7	8	7	0.47	7	3.90	8	6645.98
10_60_6	60	12	13	13	0.07	13	2.62	13	1.80
10_60_9	60	8	9	8	0.55	8	4.06	9	9757.99
10_80_1	80	12	13	12	0.88	12	8.34	13	4448.28
10_80_4	80	13	14	14	0.18	14	5.89	14	9311.99
10_80_10	80	14	15	14	0.34	14	7.07	15	2916.16
10_100_3	100	15	16	15	1.04	15	16.99	16	2525.43
10_100_5	100	17	18	17	1.06	17	16.00	18	2504.44
10_100_7	100	13	14	13	1.66	13	17.66	14	3955.68

4 ... To 2-Dimensional Knapsack

In this section we apply the methods of the previous section in order to compute upper bounds for 2KP. As mentioned in the introduction we would like to find a tight relaxation of the 2KP polytope and optimize over it. For instance, the method proposed in [8], works with the following relaxation of $P_{2\mathrm{KP}}(R)$:

$$P^0(R) := \mathrm{conv}\{x \in \{0,1\}^n : \textstyle\sum_{j=1}^n w_j h_j x_j \leq 1\}. \tag{12}$$

In the direction of defining tighter relaxations, the following corollary of Theorem 2 is pointed out in [11].

Corollary 2. *Given $R = (n, w, h) \in \mathcal{R}$, the following class of inequalities is valid for $P_{2\mathrm{UKP}}(R)$ and $P_{2\mathrm{KP}}(R)$:*

$$\sum_{j=1}^n f(w_j) g(h_j) x_j \leq 1, \quad f(\cdot), g(\cdot) \in \mathcal{D}. \tag{13}$$

The stronger result that we obtain by considering Theorem 3 instead of Theorem 2 is:

Theorem 4. *Given $R = (n, w, h) \in \mathcal{R}$, the following class of inequalities is valid for $P_{2\mathrm{KP}}(R)$:*

$$\sum_{j=1}^n \alpha_j \beta_j x_j \leq 1, \quad \begin{array}{l} \sum_{j=1}^n \alpha_j x_j \leq 1 \text{ valid for } P_{\mathrm{KP}}(I^w), \\ \sum_{j=1}^n \beta_j x_j \leq 1 \text{ valid for } P_{\mathrm{KP}}(I^h). \end{array} \tag{14}$$

In [11] the authors propose to compute a lower bound by optimizing over the convex hull of binary vectors that satisfy the inequalities (13) generated by choosing $f(\cdot), g(\cdot) \in \mathcal{D}$ out of a small number of predefined possibilities. As before, we discuss how to impose all the inequalities. We state the result for (14), the counterpart for (13) being immediate with 2DP instead of 2SDP:

Theorem 5. *Given $R = (n, w, h) \in \mathcal{R}$ and $x^* \in [0,1]^n$, the optimal solution of 2SDP with the modified objective function $\max \sum_{j=1}^n x_j^* \pi_j \rho_j$ is at most 1 if and only if x^* satisfies all inequalities (14). If this is not the case, the optimal 2SDP solution (π^*, ρ^*) defines the inequality (14) $\sum_{j=1}^n \pi_j^* \rho_j^* x_j \leq 1$ which is most violated by x^*.*

Since, as in the 1-dimensional case, our solution approach to 2DP and 2SDP is valid also for generic nonnegative objective function coefficients, using the implementation in Section 3 we can separate over constraints (13) and (14). Given a 2KP instance associated with $R \in \mathcal{R}$ and profit vector p, this allows us to optimize $p^T x$ over the following relaxations of $P_{2\mathrm{KP}}(R)$, in increasing order of strength:

$$\begin{array}{l} P^1(R) := \{x \in \mathbb{R}_+^n : (13)\}, \\ P^2(R) := \mathrm{conv}\{x \in \{0,1\}^n : (13)\}, \\ P^3(R) := \mathrm{conv}\{x \in \{0,1\}^n : (14)\}, \end{array} \tag{15}$$

Table 2. 2KP, instances from the literature

$(R,\ )$		$\mathrm{opt}_{2\mathrm{KP}}(R,\ )$	$\mathrm{opt}(P^0(R))$	$\mathrm{opt}(P^1(R))$	$\mathrm{opt}(P^2(R))$	$\mathrm{opt}(P^3(R))$
cgcut1	16	244	260	244	244	244
cgcut2	23	2892	2919	2909	2903	2903
cgcut3	62	1860	2020	2005	1940	1900
ngcut1	10	164	201	206	201	164
ngcut2	17	230	253	257	253	230
ngcut3	21	247	266	260	254	247
ngcut4	7	268	275	271	268	268
ngcut5	14	358	373	359	358	358
ngcut6	15	289	317	317	317	289
ngcut7	8	430	430	430	430	430
ngcut8	13	834	938	906	906	856
ngcut9	18	924	962	945	924	924
ngcut10	13	1452	1517	1511	1452	1452
ngcut11	15	1688	1864	1779	1726	1688
ngcut12	22	1865	2012	1953	1913	1865
gcut1	10	48368	62500	50960	48440	48368
gcut2	20	59798	62500	61630	60662	59798
gcut3	30	61275	62500	62043	61275	61275
gcut5	10	195582	250000	229301	223327	195582
gcut6	20	236305	250000	244022	237682	236305
gcut7	30	240143	250000	245300	240143	240143
gcut9	10	939600	1000000	949424	939600	939600
gcut10	20	937349	1000000	975890	947385	937349
gcut11	30	969709	1000000	988334	972140	969709
hccut1	15	5827	7016	5827	5827	5827
hccut2	7	1178	1347	1201	1178	1178
hccut3	15	1270	1547	1291	1270	1270
hccut4	10	2517	3079	2578	2517	2517
hccut5	15	2949	3604	2949	2949	2949
wang20	42	2726	2800	2800	2747	2726
okp1	50	27718	29133	28714	28431	27796
okp2	30	22502	24800	24849	24738	22906
okp3	30	24019	26714	26520	25884	24019
okp4	61	32893	33631	33246	32893	32893
okp5	97	27923	29045	28371	27934	27923

using a cutting plane approach in the first case and branch-and-cut in the other two cases.

Most questions about the worst-case strength of the above relaxations are hard and remain open. As far as we know, the only published result [8] concerns 2KP and reads:

Proposition 6. $\sup_{R\in\mathcal{R},p\in\mathbb{R}^n_+} \dfrac{\max\{p^Tx : x \in P^0(R)\}}{\mathrm{opt}_{2\mathrm{KP}}(R,p)} = 3.$

For 2UKP, we can prove the following counterpart of Proposition 3:

Proposition 7. $\sup_{R\in\mathcal{R},p\in\mathbb{R}^n_+} \dfrac{\max\{p^T x : x \in P^1(R)\}}{\mathrm{opt}_{2\mathrm{UKP}}(R,p)} \in [9/8, \zeta].$

4.1 Computational Results

In this section we present the optimal values of the relaxations in (15) for the 2KP instances in the literature. As our current implementation is very rough, we simply report the optimal values in Table 2, comparing them with $\mathrm{opt}_{2\mathrm{KP}}(R,p)$, that is known for all the instances considered [12, 8], and with the simple upper bound obtained by optimization over $P^0(R)$ as defined in (12).

The clear outcome of the results is that the optimum over $P^3(R)$ is equal to $\mathrm{opt}_{2\mathrm{KP}}(R,p)$ in almost all cases, namely for 30 out of 35 instances, with an average percentage gap of 0.2%. The same property is not shared by the other three relaxations, although the optimum over $P^2(R)$ is equal to $\mathrm{opt}_{2\mathrm{KP}}(R,p)$ in 15 cases, with an average percentage gap of 2.9%, and the optimum over $P^1(R)$, with an average percentage gap of 4.6%, is often much better than the optimum over $P^0(R)$, with an average percentage gap of 9.6%. Moreover, for the 30 instances for which the optimum over $P^3(R)$ turns out to be equal to $\mathrm{opt}_{2\mathrm{KP}}(R)$, in 27 cases the corresponding solution turns out to be feasible for 2KP.

These results suggest the development of exact algorithms based on relaxation $P^3(R)$ rather than $P^0(R)$ as the current state-of-the-art ones.

Acknowledgments. We are grateful to Charles Audet for having provided us with the BLP code described in [3, 1], to Marco Boschetti for having provided us with instance-by-instance 2BP results from [5, 6], and to Sándor Fekete for helpful discussions and for having provided us with the code used in [12] to test if a set of rectangles fits in a square.

References

1. S. Alarie , C. Audet, B. Jaumard, and G. Savard. Concavity cuts for disjoint bilinear programming. *Mathematical Programming*, 90:373–398, 2001.
2. F. A. Al-Khayyal. Generalized bilinear programming: Part I. Models, applications and linear programming relaxation. *European Journal of Operational Research*, 60:306-314, 1992.
3. C. Audet, P. Hansen, B. Jaumard, and G. Savard. A symmetrical linear maxmin approach to disjoint bilinear programming. *Mathematical Programming*, 85:573–592, 1999.
4. J. O. Berkey and P. Y. Wang. Two dimensional finite bin packing algorithms. *Journal of the Operational Research Society*, 38:423–429, 1987.
5. M.A. Boschetti and A. Mingozzi. The two-dimensional finite bin packing problem. Part I: New lower bounds for the oriented case. *4OR*, 1:27–42, 2003.

6. M.A. Boschetti and A. Mingozzi. Two-dimensional finite bin packing problems. Part II: New lower and upper bounds. *4OR*, 2:135–148, 2003.
7. A. Caprara. Packing 2-dimensional bins in harmony. In *Proceedings of the 43-rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, pages 490–499, 2002.
8. A. Caprara and M. Monaci. On the 2-dimensional knapsack problem. *Operations Research Letters*, 32:5–14, 2004.
9. S. P. Fekete and J. Schepers. New classes of fast lower bounds for bin packing problems. *Mathematical Programming*, 91:11–31, 2001.
10. S.P. Fekete, J. Schepers. A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29:353–368, 2004.
11. S.P. Fekete, J. Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60:311–329, 2004.
12. S.P. Fekete, J. Schepers. An exact algorithm for higher-dimensional orthogonal packing. Technical Report, Technische Universität Braunschweig, 2004. Available at `http://www.math.tu-bs.de/ fekete/publications.html`.
13. W. Fernandez de la Vega and G.S. Lueker. Bin packing can be solved within $1+$ in linear time. *Combinatorica*, 1:349–355, 1981.
14. C. A. Floudas and V. Visweswaran. Quadratic optimization. In R. Horst and P. M. Pardalos (eds.), *Handbook of Global Optimization*, Kluwer Academic Publishers, 1995.
15. P. C. Gilmore and R. E. Gomory. Multistage cutting problems of two and more dimensions. *Operations Research*, 13:94–119, 1965.
16. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.
17. C. C. Lee and D. T. Lee. A simple on-line bin packing algorithm. *Journal of the ACM* 32:562–572, 1985.
18. N. Karmarkar and R.M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23-rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 1982)*, pages 312–320, 1982.
19. S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations Research* 48:256–267, 2000.
20. S. Martello and D. Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management Science* 44:388–399, 1998.
21. G. P. McCormick. Computability of global solutions to factorable nonconvex programs – Part I – Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.
22. M. Monaci and P. Toth. A set-covering based heuristic approach for bin-packing problems. To appear in *INFORMS Journal on Computing*, 2005.
23. D. Pisinger and M.M. Sigurd. On using decomposition techniques and constraint programming for solving the two-dimensional bin packing problem. DIKU-rapport 03/01, Department of Computer Science, University of Copenhagen, 2003. Available at `http://www2.adm.ku.dk/publikationer/publ/5100`.
24. S. S. Seiden and R. van Stee. New bounds for multi-dimensional packing. *Algorithmica*, 36:261–293, 2003.

On the L_∞-Norm of Extreme Points for Crossing Supermodular Directed Network LPs

Harold N. Gabow

University of Colorado at Boulder, Boulder, CO 80309, USA
hal@cs.colorado.edu
http://www.cs.colorado.edu/~hal

Abstract. We discuss extensions of Jain's framework for network design [7] that go beyond undirected graphs. The main problem is approximating a minimum cost set of directed edges that covers a crossing supermodular function. We show that iterated rounding gives a factor 3 approximation, where factor 4 was previously known and factor 2 was conjectured. Our bound is tight for the simplest interpretation of iterated rounding. We also show that (the simplest version of) iterated rounding has unbounded approximation ratio when the problem is extended to mixed graphs.

1 Introduction

Iterated rounding, introduced by Jain [7], has taken its place as a general technique for approximating solutions to integer linear programs. Jain's original framework was for undirected network design problems. This paper investigates the directed case. The approximation guarantee for the directed case was known to be at most 4 but conjectured to be 2, just like the undirected case. We prove an upper bound of 3. We prove a matching lower bound, at least for the simplest version of iterated rounding. Specifically, we show that 1/3 is the tight lower bound on the L_∞-norm of an extreme point of the directed cut-covering LP for a crossing supermodular requirement function.

Previous Work. Jain showed that iterated rounding achieves a factor 2 approximation for the undirected Steiner network problem [7, 12]. Iterated rounding remains the only way known to achieve any $O(1)$-approximation for this problem. More generally Jain gave a factor 2 approximation for any undirected network design problem whose cut requirements are given by a weakly supermodular function.

Subsequently a number of iterated rounding algorithms were presented for approximating a minimum cost subgraph that satisfies a given requirement on vertex connectivity. The algorithms are all based on the setpair-covering LP. Fleischer et.al. achieve a 2-approximation for element connectivity as well as $\{0, 1, 2\}$ vertex connectivity, both on undirected graphs [5]. These are the best results known for these problems. Cheriyan et.al. [1] give an $O(\sqrt{m})$-approximation for

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 392–406, 2005.

setpair covering, which extends to an $O(n/\sqrt{n-k})$-approximation for k-vertex connectivity on both directed and undirected graphs [2] (a recent improvement is given in [9]).

Detouring a little, let *k-EC* stand for "k-edge connected" and let *k-ECSS* stand for "k-edge connected spanning subgraph". A problem of interest is finding a minimum cost k-ECSS. We will also refer to the problem of finding a minimum cost k-EC augmentation (this is the same problem, except the cost 0 edges are not recorded as part of the solution). Khuller and Vishkin give 2-approximations for these two problems, on both directed and undirected graphs [8]. Their work generalizes the algorithm for the case $k = 1$ by Frederickson and Ja'Ja' [4].

Most relevant to our results is the work of Melkonian and Tardos [10]. They generalize Jain's framework to directed graphs. Jain's weakly supermodular cut requirement function becomes a crossing supermodular function. This is slightly less powerful, but the directed model still includes problems of interest. For example the model includes the minimum cost directed k-ECSS problem, and more sophisticated versions of this problem such as making a directed hypergraph k-edge connected by adding a minimum cost set of directed graph edges. (See Section 2.)

Melkonian and Tardos [10] proved that iterated rounding achieves an approximation ratio of 4. They conjectured that the true approximation ratio is 2 (like the undirected case) and supported this conjecture with experimental evidence.

In addition to the iterated rounding algorithm, [10] proves an approximation ratio of 2 for their problem by using a different algorithm: It consists of two applications of an algorithm of Frank [3] which gives an exact solution in the special case of an intersecting supermodular cut requirement function. The resulting approximation algorithm is a natural generalization of [4, 8].

In the experiments this 2-approximation gave poorer results than iterated rounding, e.g., the 2-approximation averaged 25% above optimum while iterated rounding averaged within 1.5% of optimum (more details are in [10]). Such results, plus the success of iterated rounding on other problems, motivate our further study of the directed case.

Our Results. The only known method of proving a performance bound for iterated rounding is via Jain's original result. It says iterated rounding gives a $1/\epsilon$-approximation when every optimum extreme point of the LP family has L_∞-norm at least ϵ (for $0 < \epsilon < 1$). [10] conjectured that supermodular directed cut problems have $\epsilon = 1/2$. We prove a tight bound of $\epsilon = 1/3$. More precisely for the cut-covering LP of any supermodular directed cut problem, every nonzero extreme point has L_∞-norm at least $1/3$. Furthermore the simplest supermodular directed cut problem of interest, minimum cost k-EC augmentation, has an infinite family of instances where the unique optimum extreme point is $(1/3, \ldots, 1/3)$, the vector with all coordinates equal to $1/3$. (This holds for any integer $k \geq 1$. The proof is based on a "generic" example which should allow it to extend to other supermodular cut problems.) The simplest version of iterated rounding gives approximation ratio exactly 3 on these instances (for both minimum cost k-EC augmentation and minimum cost k-ECSS). We leave open the

possibility that a more sophisticated rounding strategy achieves approximation ratio 2. However new proof techniques will be required for such a result.

The reason we can prove a tight lower bound is that we concentrate more on linear independence than on graph properties.

The last part of this paper briefly examines the supermodular cut problem on mixed graphs. Here the solution may use both directed and undirected edges. The main finding is negative: Looking again at minimum cost k-EC augmentation, for any integer $k > 1$ there is an infinite family of instances where the unique optimum extreme point is $(2/n, \ldots, 2/n)$, for n the number of vertices. But for minimum cost 1-EC augmentation and its generalizations like the hypergraph augmentation problem mentioned above, 1/4 is the tight lower bound for the L_∞-norm of an extreme point of the cut-covering LP. So iterated rounding is a 4-approximation for these problems.

Section 2 defines the supermodular directed cut problem and gives several examples. Section 3 proves the L_∞ lower bound, implying that iterated rounding achieves approximation ratio 3. Section 4, showing that our bound is tight, and Section 5, the mixed graph results, are much abbreviated. The complete paper is available as [6].

Terminology. Throughout this paper n and m denote the number of vertices and number of edges of the given graph $G = (V, E)$, respectively. In all graphs, each edge e has an associated integral multiplicity that specifies the number of copies of e. (m does not count multiplicities.) Self-loops are not allowed. We also refer to functions that assign each edge e a weight, which plays the role of the number of copies of e but need not be integral.

For $S, T \subseteq V$, $\delta(S,T)$ denotes the number of edges directed from $S - T$ to $T - S$. In addition we define $\delta(S) = \delta(S, V - S)$, $\rho(S,T) = \delta(T,S)$ and $\rho(S) = \rho(S, V - S)$.

A set $S \subseteq V$ is *k-edge connected* if no two vertices of S are separated by a cut of $< k$ edges, i.e., any set of vertices X with $S \cap X, S - X \neq \emptyset$ has $\rho(X) \geq k$. Equivalently any two distinct vertices $s, t \in S$ are joined by k edge-disjoint st-paths.

2 Crossing Supermodular Directed Cut Problems

Sets $S, T \subseteq V$ *cross* if each of the four sets $S - T, T - S, S \cap T$ and $V - (S \cup T)$ is nonempty. A family $\mathcal{C}$ of subsets of V is *crossing* if $S, T \in \mathcal{C}$ with S and T crossing implies $S \cap T, S \cup T \in \mathcal{C}$. In this case a function $f : \mathcal{C} \to \Re$ is *crossing supermodular* if $f(S) + f(T) \leq f(S \cap T) + f(S \cup T)$ for every two crossing sets $S, T \in \mathcal{C}$.

Let $G = (V, E)$ be the given graph. We use the vector space $\Re^m$, where each dimension corresponds to an edge of E. In particular we use this notation for vectors:

$\boldsymbol{x}$ is a vector of values x_e for each $e \in E$. For $F \subseteq E$, $\boldsymbol{x}_F$ denotes the vector $\boldsymbol{x}$ with every component outside of F zeroed.

For $e \in E$, $\boldsymbol{e}$ denotes the unit vector having $x_e = 1$.

For $S \subseteq V$, $\boldsymbol{\delta}(S)$ ($\boldsymbol{\rho}(S)$) denotes the *boundary vector* of edges leaving (entering) S, i.e., the entry for e is 1 if e leaves (enters) S, and 0 otherwise.

For $r \in \Re$, $(r, \ldots, r)$ denotes the vector of $\Re^m$ with every component equal to r.

A *(crossing) supermodular (directed) cut problem* is specified by a digraph $G = (V, E)$, a crossing family $\mathcal{C} \subseteq 2^V$ with crossing supermodular function $f : \mathcal{C} \to \mathbf{Z}$, and three vectors in $\Re^m_+$, an edge cost vector $\boldsymbol{c}$ and upper and lower bound vectors $\boldsymbol{u}$ and $\boldsymbol{l}$ respectively. Without loss of generality we assume $\boldsymbol{l} \leq \boldsymbol{u}$, $\boldsymbol{l}$ and $\boldsymbol{u}$ are integral vectors, and we allow infinite upper bounds. We seek a minimum cost set of edges respecting the upper and lower bounds such that every set $S \in \mathcal{C}$ has in-degree at least $f(S)$.

We list several supermodular cut problems that are of interest. The simplest is finding a minimum cost directed k-ECSS ($\mathcal{C} = 2^V - \{\emptyset, V\}$, $f \equiv k$).

(a) Consider a "host" graph $H = (W, F)$, a subset of vertices $V \subseteq W$, and a set of "candidate" edges $E \subseteq V \times V$. A *k-EC internal augmentation (of V)* is a subset $E' \subseteq E$ with V k-edge connected in the graph $(W, E' \cup F)$. (Here E' and F are multisets.) The problem is to find a minimum cost k-EC internal augmentation of V, given directed graph H, vertex set V, plus candidate directed edges E and their costs.

(b) (This is a special case of (a).) Recall that in a directed hypergraph H, each hyperedge consists of a set of head vertices and a set of tail vertices [11–p.769]. A *graph-edge augmentation* of H consists of H plus a number of (ordinary graph) edges. A *k-EC graph-edge augmentation* makes H k-edge connnected. The problem is to find such an augmentation of minimum cost, given H plus a set of (ordinary) candidate directed edges and their costs.

(c) In any supermodular cut problem we can add arbitrary lower bound constraints on the in-degree and out-degree of each vertex. For instance consider the problem of finding a minimum cost spanning out-tree directed from a given root vertex. Adding lower bounds on the out-degree of each vertex gives a supermodular cut problem that is NP-hard (it contains the directed Hamiltonian path problem).

(d) More generally the given edges may be colored red or green and we can require some minimum number of edges of each color in each direction at each vertex. Still more generally we can require the solution to contain at least $i_v(S)$ ($o_v(S)$) edges of S for every $S \subseteq \mathcal{I}_v$ ($S \subseteq \mathcal{O}_v$) and every $v \in V$, where $\mathcal{I}_v$ is an intersecting family of subsets of the edges directed to v, $i_v : \mathcal{I}_v \to \mathbf{Z}$ is an intersecting supermodular function, and $\mathcal{O}_v$, o_v are defined similarly on the edges directed from v.

We use the following "cut-covering" linear program, which results from dropping the integrality constraints from the supermodular directed cut problem. Recall our conventions for vectors, e.g., the inner product $\boldsymbol{x} \cdot \boldsymbol{\rho}(S)$ equals the in-degree of S using edge weights given by $\boldsymbol{x}$.

$$(LP) \qquad \begin{array}{lllll} \text{minimize} & \boldsymbol{x}\cdot\boldsymbol{c} & & & \\ \text{subject to} & \boldsymbol{x}\cdot\boldsymbol{\rho}(S) & \geq & f(S) & S\in\mathcal{C} \\ & \boldsymbol{x} & \leq & \boldsymbol{u} & \\ & \boldsymbol{x} & \geq & \boldsymbol{l} & \end{array}$$

Our approximation algorithm is iterated rounding. Although any version suffices for our upper bounds we need to be specific for the lower bound examples. We use *round-the-highest* [10]: Find an optimum extreme point $\boldsymbol{x}$. Fix the value of all components x_e that are largest (i.e., equal to $\|\boldsymbol{x}\|_\infty$) to what will be their final value, $\lceil x_e\rceil$. Modify (LP) to the residual problem (i.e., for each newly fixed e, decrease $f(S)$ by x_e for each S entered by e, and then discard e from the problem). Repeat the procedure until all values are fixed.

As shown in [7, 12], if any nonzero extreme point for any instance of (LP) has L_∞-norm at least ϵ $(0<\epsilon<1)$ then iterated rounding achieves a $1/\epsilon$ approximation ratio.

3 Lower Bound Analysis

This section is devoted to a proof of our L_∞ lower bound:

Theorem 1. *Any nonzero extreme point to (LP) has L_∞-norm at least $1/3$. Hence iterated rounding achieves approximation ratio 3 for the supermodular directed cut problem.*

Consider an arbitrary extreme point $\boldsymbol{x}$ to (LP). By definition the m components of $\boldsymbol{x}$ are uniquely determined by some set of m constraints that have been changed to equalities. Let F be the set of edges that are not determined by their bounds, $F=\{e : x_e\notin\{l_e,u_e\}\}$. The vector $\boldsymbol{x}_F$ is determined by the degree constraints. Melkonian and Tardos showed these constraints come from a laminar family:

Lemma 1 ([10]). *There is a laminar family of sets $\mathcal{L}=\mathcal{I}\cup\mathcal{O}$ such that $\boldsymbol{x}_F$ is the unique solution for $\boldsymbol{y}_F$ in the linear system*

$$\begin{array}{ll} \boldsymbol{y}_F\cdot\boldsymbol{\rho}(S)=i(S) & S\in\mathcal{I} \\ \boldsymbol{y}_F\cdot\boldsymbol{\delta}(S)=o(S) & S\in\mathcal{O} \end{array} \tag{1}$$

where each right-hand side $i(S), o(S)$ is a positive integer.

It may help to remind the reader how the system (1) comes about: $\boldsymbol{x}_F$ is determined by degree constraints of (LP), $\boldsymbol{x}\cdot\boldsymbol{\rho}(S)=f(S)$, when we fix the values $x_e, e\notin F$ to their appropriate bound l_e, u_e. This directly gives the constraint of (1) for a set $S\in\mathcal{I}$. Thus $i(S)$ is $f(S)$ decreased by the fixed (integral) values x_e with e entering S. The constraint for a set $S\in\mathcal{O}$ comes from the equation for $V-S$, and $o(S)$ is $f(V-S)$ decreased by the fixed values x_e with e leaving S. We note that a set may belong to both $\mathcal{I}$ and $\mathcal{O}$. Finally in applying Lemma 1 we will assume the system (1) is minimal. So the number of constraints

equals the number of unknowns, i.e., $|\mathcal{I}| + |\mathcal{O}| = |F|$, and the constraint vectors $\{\boldsymbol{\rho}(S) : S \in \mathcal{I}\} \cup \{\boldsymbol{\delta}(S) : S \in \mathcal{O}\}$ are linearly independent.

We will use the terms *$\mathcal{I}$-set* and *$\mathcal{O}$-set* with the obvious meaning. Now we define a rooted forest $\mathcal{F}$ that represents the laminar family $\mathcal{L}$. The nodes of $\mathcal{F}$ are partitioned into two sets $\mathcal{I}_\mathcal{F}$ and $\mathcal{O}_\mathcal{F}$, representing the $\mathcal{I}$-sets and $\mathcal{O}$-sets of $\mathcal{L}$ respectively. Thus $\mathcal{F}$ has $|\mathcal{I}| + |\mathcal{O}|$ nodes. A set belonging to $\mathcal{I} \cap \mathcal{O}$ gives rise to two nodes.

Throughout the discussion we identify each forest node and its set. This may cause slight ambiguities but they will be resolved from the context. For instance if S is a node of $\mathcal{F}$, saying $S \in \mathcal{I}_\mathcal{F}$ means that S is being treated as a node while saying $S \in \mathcal{I}$ means S is being treated as a set.

If S is a set in $\mathcal{I} \cap \mathcal{O}$ let S_p and S_c denote its two corresponding nodes. Define S_p to be the parent of S_c in $\mathcal{F}$. It is irrelevant which of the nodes S_p, S_c belongs to $\mathcal{I}_\mathcal{F}$ and which belongs to $\mathcal{O}_\mathcal{F}$.

We complete the definition of $\mathcal{F}$ by specifying the remaining parent-child relations: The children of a node S are the nodes T corresponding to the maximal proper subsets of S in $\mathcal{L}$. In this definition replace S by S_c if $S \in \mathcal{I} \cap \mathcal{O}$, and similarly replace T by T_p if $T \in \mathcal{I} \cap \mathcal{O}$.

A node of $\mathcal{F}$ is a *leaf*, *chain node* or *branching node* depending on whether it has 0, 1 or > 1 children, respectively. Chain nodes have a further classification: A chain node is a *1-node* if it belongs to the same family $\mathcal{I}_\mathcal{F}$, $\mathcal{O}_\mathcal{F}$ as its (unique) child. If not (its child belongs to the opposite family) it is a *2-node*. For example if $S \in \mathcal{I} \cap \mathcal{O}$ then S_p is necessarily a 2-node; S_c may be any type of node.

The overall proof strategy is Jain's [7, 12]: Assume for the sake of contradiction that no edge has value $\geq 1/3$, i.e., $0 < x_e < 1/3$ for every edge e. (We can discard edges with $x_e = 0$ from the problem.) We will assign two ends of an edge to each node of $\mathcal{F}$. Each end will be assigned at most once, and at least two ends will remain unassigned. This contradicts the fact that the number of edges equals the number of nodes of $\mathcal{F}$ (by Lemma 1).

We will define a notion of "availability" to guide our assignment of ends. Each node of $\mathcal{F}$ will be assigned ends that are available to it. Intuitively we will show that ends available to the leaves of $\mathcal{F}$ pay for the leaves and the branching nodes, while ends available to the 1-nodes pay for themselves. The 2-nodes pay for themselves, but their available ends can come from other 2-nodes or from branching nodes. The most involved part of the argument is the analysis of 2-nodes. For this reason we postpone all details concerning 2-nodes and begin by discussing the other types of nodes.

Consider a chain node S with unique child A. Let e be an edge with an end in $S - A$. We'll call e "p-directed" (for "parent-directed") if it's oriented consistent with S's family ($\mathcal{I}$ or $\mathcal{O}$) and "c-directed" (for "child-directed") if it's oriented consistent with A's family. (See Fig.1.) In precise terms e is *p-directed* if either

(i) $S \in \mathcal{I}_\mathcal{F}$ and e enters S or A, or
(ii) $S \in \mathcal{O}_\mathcal{F}$ and e leaves S or A.

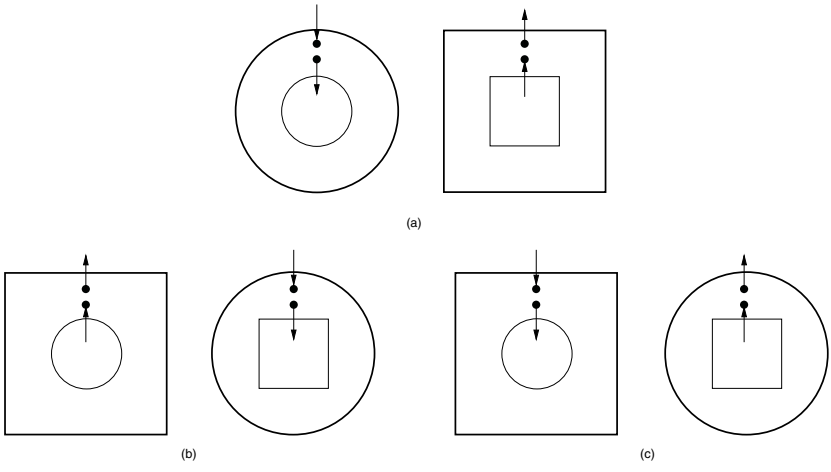


Fig. 1. Chain nodes with their unique child. Circles represent $\mathcal{I}$-sets and squares represent $\mathcal{O}$-sets. The chain node is outermost and drawn heavy. (a) 1-nodes and their available edge ends. The edges are both p-directed and c-directed. (b)–(c) 2-nodes and their corresponding available edges, when the 2-node is (b) the root, (c) a leaf. In (b) the edges are p-directed and in (c) the edges are c-directed

Similarly e is *c-directed* if either

(i) $A \in \mathcal{I}_{\mathcal{F}}$ and e enters S or A, or
(ii) $A \in \mathcal{O}_{\mathcal{F}}$ and e leaves S or A.

Note that in (i) only one of the two possibilities holds, i.e., e cannot enter both S and A, since it has an end in $S - A$. Similarly in (ii). Also for a 1-node the notions of p-directed and c-directed are identical, while for a 2-node they are opposites.

Let vertex v be an end of edge e and let S be a node of $\mathcal{F}$. Below we define when v, as an end of e, is available to S. For simplicity we will abbreviate this to "v is available to S" except when it is necessary to refer to e. The definition for 1-nodes is illustrated in Fig.1(a). Availability to 2-nodes will be defined later, here we give the definition for all the other cases.

Definition 1. *End v of edge e is* available *to node S of $\mathcal{F}$ if (i) or (ii) holds:*

(i) When S is a leaf, $v \in S$, and e is directed consistent with S's family, i.e., either $S \in \mathcal{I}_{\mathcal{F}}$ and e enters S or $S \in \mathcal{O}_{\mathcal{F}}$ and e leaves S.

(ii) When S is a 1-node, $v \in S - A$ for A the child of S, and e is p-directed (equivalently, c-directed).

Clearly laminarity implies that v, as an end of e, is available to at most one node.

Lemma 2. *The number of ends available to leaves of $\mathcal{F}$ exceeds twice the number of leaves and branching nodes.*

Proof. A leaf $S \in \mathcal{I}_\mathcal{F}$ is entered by ≥ 4 fractional edges. (If not, $\boldsymbol{x}\cdot\boldsymbol{\rho}(S)$ a positive integer makes $x_e \geq 1/3$ for some entering edge e.) Let $\mathcal{F}$ have ℓ leaves. Hence $\geq 4\ell$ edge ends are available to the leaves. $\mathcal{F}$ has $< \ell$ branching nodes, since any rooted forest has fewer branching nodes than leaves. The lemma amounts to $4\ell = 2(2\ell)$.

The rest of the proof is based on the linear independence of the vectors $\{\boldsymbol{\rho}(S) : S \in \mathcal{I}\} \cup \{\boldsymbol{\delta}(S) : S \in \mathcal{O}\}$ (Lemma 1).

Lemma 3. *Each 1-node has at least two available ends.*

Proof. Consider a 1-node S with child A, where *wlog* $S, A \in \mathcal{I}_\mathcal{F}$. The vector $\boldsymbol{a} = \boldsymbol{\rho}(S) - \boldsymbol{\rho}(A)$ has entries $0, \pm 1$. Every edge e with $a_e = \pm 1$ has an end available to S: If $a_e = 1$ then e enters S but not A. If $a_e = -1$ then e enters A but not S. In both cases e is p-directed and has an end in $S - A$ available to S.

Now we need only show $\boldsymbol{a}$ has ≥ 2 nonzero entries. $\boldsymbol{a} \neq \boldsymbol{0}$ by linear independence. $\boldsymbol{x}\cdot\boldsymbol{a}$ is a signed sum of fractions x_e that is an integer (Lemma 1) so there cannot be only 1 nonzero a_e.

We turn to the 2-nodes. From Lemmas 2–3 it suffices to show the 2-nodes pay for themselves. We begin with two simple facts that form the core of the proof:

Proposition 1. *(i) For any sets* $A \subseteq S \subseteq V$, $\boldsymbol{\delta}(S) = \boldsymbol{\delta}(A) + \boldsymbol{\delta}(S - A, V - S) - \boldsymbol{\delta}(A, S - A)$. *Similarly for* $\boldsymbol{\rho}(S)$.

(ii) For any $S \subseteq V$ *partitioned into sets* A_i, $i = 1, \ldots, r$, $\boldsymbol{\delta}(S) - \boldsymbol{\rho}(S) = \sum_{i=1}^{r} \boldsymbol{\delta}(A_i) - \boldsymbol{\rho}(A_i)$.

For each node S of $\mathcal{F}$ define $\mathcal{F}_S$ to be the minimal subtree of $\mathcal{F}$ having S as its root and each leaf either a leaf of $\mathcal{F}$ or a 2-node. As a simple example, $\mathcal{F}_S$ is the single node S when S is a leaf or a 2-node.

For each 2-node S we define a second subtree $\mathcal{F}^*_S$. $\mathcal{F}^*_S$ is defined like $\mathcal{F}_S$ except S must be interior. More precisely $\mathcal{F}^*_S$ is the minimal subtree of $\mathcal{F}$ having S as its root and each leaf either a leaf of $\mathcal{F}$ or a 2-node other than S.

The various trees $\mathcal{F}^*_S$ can overlap: A 2-node S occurs at the root in $\mathcal{F}^*_S$, and also as a leaf in $\mathcal{F}^*_T$ for T the first 2-node that is a proper ancestor of S (if such an ancestor exists). It is easy to see these are the only possible overlaps.

We now complete the definition of availability. See Fig.1(b)–(c).

Definition 2. *End* v *of edge* e *is* available *to the 2-node* S *if*

(i) v *(as the end of* e*) is not available to any leaf or 1-node;*

(ii) the deepest node of $\mathcal{F}$ *that contains* v*, say* X*, belongs to* $\mathcal{F}^*_S$*;*

(iii) if X *is a 2-node then either* $X = S$ *and* e *is p-directed, or* $X \neq S$ *and* e *is c-directed.*

An end v of edge e is available to at most one node S of $\mathcal{F}$. This is clear unless X belongs to two trees $\mathcal{F}^*_S$. In that case X is a 2-node and the trees are $\mathcal{F}^*_X$ and $\mathcal{F}^*_S$ for some $S \neq X$. Since X is a 2-node edge e cannot be both p-directed and c-directed (as in Fig.1(b)–(c)). Hence (*iii*) makes v available to only one of X and S.

The analysis of 2-nodes uses Proposition 1 to derive a nontrivial linear equation for $\boldsymbol{\rho}(S)$ ($\boldsymbol{\delta}(S)$) when S is a 2-node in $\mathcal{I}_{\mathcal{F}}$ ($\mathcal{O}_{\mathcal{F}}$). We call this equation the "r-expansion" of $\boldsymbol{\rho}(S)$ ($\boldsymbol{\delta}(S)$), since it is used at the root of $\mathcal{F}_S^*$. The r-expansion will immediately give the two edge ends that pay for S. We derive the r-expansion from similar equations for the other nodes of $\mathcal{F}_S^*$. More precisely we use similar equations for $\boldsymbol{\delta}(S)$ ($\boldsymbol{\rho}(S)$) when S is an arbitrary node in $\mathcal{I}_{\mathcal{F}}$ ($\mathcal{O}_{\mathcal{F}}$). We call these equations the "n-expansion" of $\boldsymbol{\delta}(S)$ ($\boldsymbol{\rho}(S)$) since they are used at the nonroots of $\mathcal{F}_S^*$. Notice the switch, e.g., when S is a 2-node in $\mathcal{I}_{\mathcal{F}}$ the r-expansion is for $\boldsymbol{\rho}(S)$ and the n-expansion is for $\boldsymbol{\delta}(S)$. We will now give a precise definition of r-expansions. After that we'll give the modifications needed for n-expansions.

We use this additional notation: If $\mathcal{T}$ is a subtree of $\mathcal{F}$ then $\overline{\mathcal{T}}$ denotes the tree $\mathcal{T}$ extended by including the (unique) child of each 2-node leaf of $\mathcal{T}$. For instance Fig.1(b) illustrates $\overline{\mathcal{F}}_S$ for a 2-node S.

Definition 3. *Let S be a 2-node. If $S \in \mathcal{I}_{\mathcal{F}}$ ($\mathcal{O}_{\mathcal{F}}$) an* r-expansion for $\boldsymbol{\rho}(S)$ ($\boldsymbol{\delta}(S)$) *is an equation whose left-hand side is $\boldsymbol{\rho}(S)$ ($\boldsymbol{\delta}(S)$) and whose right-hand side is*

$$\sum_{A\in O} \sigma_A \boldsymbol{\delta}(A) + \sum_{B\in I} \sigma_B \boldsymbol{\rho}(B) + \sum_{e\in AV} \sigma_e \boldsymbol{e} \tag{2}$$

where

(i) σ_A, σ_B and σ_e are integers;
(ii) O (I) is a set of nodes contained in $\overline{\mathcal{F}}_S^* \cap \mathcal{O}_{\mathcal{F}} - S$ ($\overline{\mathcal{F}}_S^* \cap \mathcal{I}_{\mathcal{F}} - S$);
(iii) AV is a set of edges, each having an end that is available to S.

It can be proved that in our derivation all σ factors are actually just signs, $\pm$. Our main result only requires the σ's to be integral. So this paper will ignore this special property.

Example 1. Consider three chain nodes, $O, O' \in \mathcal{O}_{\mathcal{F}}$, $I \in \mathcal{I}_{\mathcal{F}}$, with I the unique child of O and O' the unique child of I. Clearly O and I are 2-nodes. It is possible that I represents the same set as O or O'. The tree $\mathcal{F}_O^*$ consists of root O and leaf I; $\overline{\mathcal{F}}_O^*$ consists of O, I and leaf O'. The r-expansion of $\boldsymbol{\delta}(O)$ will derive from Proposition 1(i) and have the form $\boldsymbol{\delta}(O) = \boldsymbol{\delta}(O') + \sum \sigma_e \boldsymbol{e}$.

To extend Definition 3 to n-expansions let S be an arbitrary node (possibly a 2-node). Let $\widehat{S}$ be the first proper ancestor of S that is a 2-node, if such exists. If $S \in \mathcal{I}_{\mathcal{F}}$ ($S \in \mathcal{O}_{\mathcal{F}}$) an *n-expansion for* $\boldsymbol{\delta}(S)$ ($\boldsymbol{\rho}(S)$) is defined just like Definition 3 changing (ii)–(iii) to the following:

(ii') $O \subseteq \overline{\mathcal{F}}_S \cap \mathcal{O}_{\mathcal{F}}$ and $I \subseteq \overline{\mathcal{F}}_S \cap \mathcal{I}_{\mathcal{F}}$;

(iii') AV is a set of edges, each having an end that is available to $\widehat{S}$.

The following is our main lemma.

Lemma 4. *Consider a node $S \in \mathcal{I}_{\mathcal{F}}$. $\boldsymbol{\delta}(S)$ has an n-expansion, and if S is a 2-node then $\boldsymbol{\rho}(S)$ has an r-expansion. Symmetrically for a node $S \in \mathcal{O}_{\mathcal{F}}$.*

Proof. The proof is by induction on the height of S in $\mathcal{F}$. The inductive assertion consists of the equation for $\boldsymbol{\delta}(S)$ or $\boldsymbol{\rho}(S)$ with right-hand side (2), plus properties

(i) –(ii) below. The two properties simply restate the remaining restrictions from Definition 3 and its extension.

The notation $\widehat{S}$ was introduced for n-expansions as the first proper 2-node ancestor of S. To unify the two cases take $\widehat{S}$ to be S for r-expansions.

Consider any vector $\boldsymbol{e}$ occuring in (2), i.e., $e \in AV$. We restate properties (iii) and (iii') of the definitions:

(i) Edge e has an end v available to $\widehat{S}$.

Next we treat (ii) of Definition 3 and (ii'). These statements are essentially syntactic and require no explicit verification, except for excluding the term $\boldsymbol{\rho}(S)$ in an r-expansion for $\boldsymbol{\rho}(S)$: In the following O and I refer to the sets of (2).

(ii) In an r-expansion for S, $S \notin O \cup I$.

We organize the induction in three cases corresponding to the node types leaf, chain, and branching respectively. We always assume that we seek an expansion (r- or n-) for $\boldsymbol{\delta}(S)$. This implies $S \in \mathcal{I}_F$ for an n-expansion and $S \in \mathcal{O}_F$ for an r-expansion. The argument for an expansion for $\boldsymbol{\rho}(S)$ is symmetric.

Case 1. S is a leaf of $\mathcal{F}$. A leaf only has an n-expansion, so this case makes $S \in \mathcal{I}_\mathcal{F}$. $\mathcal{F}_S$ consists of the single node S. The desired equation is

$$\boldsymbol{\delta}(S) = \sum \boldsymbol{e}$$

where e ranges over all edges leaving S. Property (i) holds since the tail of an edge leaving S is not available to S (as $S \in \mathcal{I}_\mathcal{F}$) and so the tail is available to $\widehat{S}$. (ii) is vacuous.

Case 2. S is a chain node. Let A be the unique child of S. We will sometimes break the analysis into three subcases. Since we seek an expansion of $\boldsymbol{\delta}(S)$ the following subcases exhaust all possibilities:

(a) S is a 1-node. So $A, S \in \mathcal{I}_\mathcal{F}$ and we seek an n-expansion. Clearly $\widehat{S} \neq S$.

(b) S is a 2-node in $\mathcal{I}_\mathcal{F}$. So $A \in \mathcal{O}_\mathcal{F}$ and again we seek an n-expansion. The subtree $\mathcal{F}_S$ consists of the single node S. Also $\widehat{S} \neq S$.

(c) S is a 2-node in $\mathcal{O}_\mathcal{F}$. So $A \in \mathcal{I}_\mathcal{F}$ and we seek an r-expansion. The subtree of interest is $\mathcal{F}^*_S$, and $\widehat{S} = S$.

Proposition 1(i) applies to any chain node S and gives

$$\boldsymbol{\delta}(S) = \boldsymbol{\delta}(A) + \sum \pm \boldsymbol{e} \tag{3}$$

where $\sum \pm \boldsymbol{e}$ has e ranging over all edges with coefficient 1 in $\boldsymbol{\delta}(S - A, V - S)$ or $\boldsymbol{\delta}(A, S - A)$.

We show the sum $\sum \pm \boldsymbol{e}$ satisfies property (i) for v chosen as the end of e in $S - A$: First suppose S is a 1-node. End v of e is not available to the 1-node S (e is not p-directed since $S \in \mathcal{I}_\mathcal{F}$). Clearly this implies v is not available to any leaf or 1-node. This makes v available to $\widehat{S}$.

If S is a 2-node then obviously v is not available to any leaf or 1-node. If $S \in \mathcal{I}_\mathcal{F}$ then $S \neq \widehat{S}$ and e is c-directed (Fig.1(c)), so v is available to $\widehat{S}$. If $S \in \mathcal{O}_\mathcal{F}$ then $S = \widehat{S}$ and e is p-directed (Fig.1(b)), so again v is available to $\widehat{S}$. This establishes (i).

We complete the analysis of Case 2 in two subcases. First suppose S is a 2-node in $\mathcal{I}_\mathcal{F}$. Then (3) is the desired n-expansion for S. The term $\boldsymbol{\delta}(A)$ is allowed in the expansion since $A \in \overline{\mathcal{F}}_S \cap \mathcal{O}_\mathcal{F}$. (If $S \in \mathcal{I} \cap \mathcal{O}$ and A represents the same set as S, our n-expansion is $\boldsymbol{\delta}(S) = \boldsymbol{\delta}(A)$, i.e., $\boldsymbol{\delta}(S) = \boldsymbol{\delta}(S)$. That's fine.)

The remaining possibilities are that S is a 1-node, or a 2-node in $\mathcal{O}_\mathcal{F}$. Both alternatives have $A \in \mathcal{I}_\mathcal{F}$. To get the desired n-expansion for $\boldsymbol{\delta}(S)$, take (3) and replace $\boldsymbol{\delta}(A)$ by its n-expansion given by induction. (This expansion exists since $A \in \mathcal{I}_\mathcal{F}$.)

Property (i) holds by induction. For (ii) suppose S is a 2-node in $\mathcal{O}_\mathcal{F}$. The right-hand side of the expansion of $\boldsymbol{\delta}(A)$ contains boundary vectors $\boldsymbol{\delta}(C)$, $\boldsymbol{\rho}(C)$ only for nodes $C \in \overline{\mathcal{F}}_A$. Clearly C is not node S. (Note that if A represents the same set as S, we have $\boldsymbol{\rho}(A) \equiv \boldsymbol{\rho}(S)$ on the right-hand side of the expansion. That's fine.)

Case 3. S is a branching node. A branching node only has an n-expansion so this case has $S \in \mathcal{I}_\mathcal{F}$. Let the children of S be A_i, $i = 1, \ldots, r$ $(r > 1)$. Each A_i may belong to $\mathcal{I}_\mathcal{F}$ or $\mathcal{O}_\mathcal{F}$. Define A_0 to be the set $S - \bigcup_{i=1}^{r} A_i$. It is possible that A_0 is empty, and of course A_0 does not belong to $\mathcal{L}$.

Proposition $1(ii)$ applied to the partition A_i, $i = 0, 1, \ldots, r$ gives

$$\boldsymbol{\delta}(S) = \boldsymbol{\rho}(S) + \sum_{i=1}^{r} \boldsymbol{\delta}(A_i) - \boldsymbol{\rho}(A_i) + \sum \pm \boldsymbol{e} \tag{4}$$

where $\sum \pm \boldsymbol{e}$ has e ranging over all edges incident to A_0. This sum satisfies (i): Let v be the end of e in A_0. Clearly v is not owned by any leaf or 1-node, so v is available to $\widehat{S}$.

To get the desired expansion for S, start with (4). Then for $i = 1, \ldots, r$, if $A_i \in \mathcal{I}_\mathcal{F}$ $(A_i \in \mathcal{O}_\mathcal{F})$ replace $\boldsymbol{\delta}(A_i)$ $(\boldsymbol{\rho}(A_i))$ by its n-expansion, respectively.

Note that property (i) continues to hold by induction. (ii) is vacuous. This completes the induction.

Observe from the proof that each vector $\boldsymbol{e}$ in an expansion has $|\sigma_e| \leq 2$. This follows since $\boldsymbol{e}$ can be introduced into the expansion at most twice, once for each of its ends.

Now we prove our goal, that each 2-node S has 2 available ends. By symmetry assume $S \in \mathcal{O}_\mathcal{F}$. In the r-expansion of $\boldsymbol{\delta}(S)$ each boundary vector is either $\boldsymbol{\delta}(C)$ for C an $\mathcal{O}$-set distinct from S, or $\boldsymbol{\rho}(C)$ for C an $\mathcal{I}$-set. Since the boundary vectors for $\mathcal{L}$ are linearly independent, the expansion involves ≥ 1 edge vector term $\boldsymbol{e}$. (If $S \in \mathcal{O}_\mathcal{F}$ corresponds to the same set of $\mathcal{L}$ as its child $A \in \mathcal{I}_\mathcal{F}$ then $\boldsymbol{\rho}(A) \equiv \boldsymbol{\rho}(S)$ may occur in the expansion. That's fine.)

Finally suppose the expansion has exactly 1 term $\boldsymbol{e}$. Multiplying the expansion by $\boldsymbol{x}$, all terms from boundary vectors (e.g., $\boldsymbol{x} \cdot \boldsymbol{\delta}(S)$) are integral. Thus

$\sigma_e x_e$ is integral. Now $|\sigma_e| \leq 2$ implies x_e is half-integral, contradicting $x_e < 1/3$. We conclude there are ≥ 2 terms $\boldsymbol{e}$. This gives S 2 available ends. The analysis of the charging scheme is complete.

4 Tight Example

This section summarizes the proof of:

Theorem 2. *For any integer $k \geq 1$, an infinite family of instances of the minimum cost directed k-EC augmentation problem has $(1/3, \ldots, 1/3)$ as the unique optimum solution to (LP). Round-the-highest has approximation ratio 3 on these instances (for both mincost k-EC augmentation and mincost k-ECSS).*

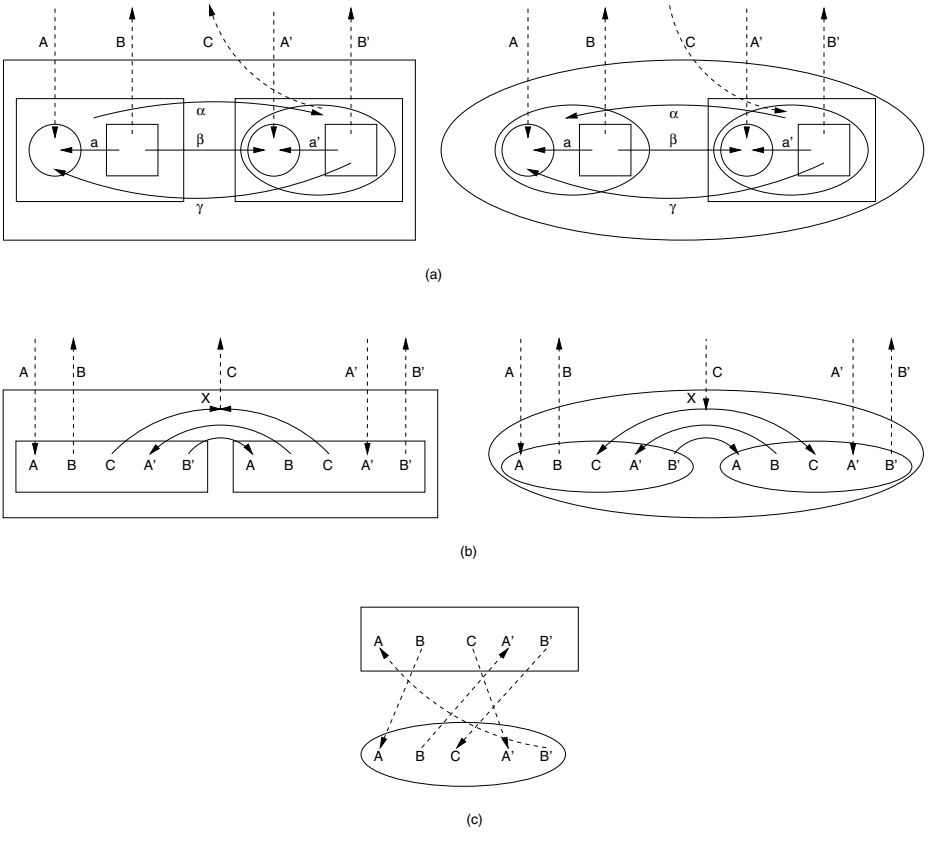


Fig. 2. Generic example. Every module has five incident edges A, B, C, A', B', all drawn dashed. (a) Elementary modules of type $\mathcal{O}$ (left) and $\mathcal{I}$ (right). (b) Compound modules of type $\mathcal{O}$ (left) and $\mathcal{I}$ (right). (c) Root modules

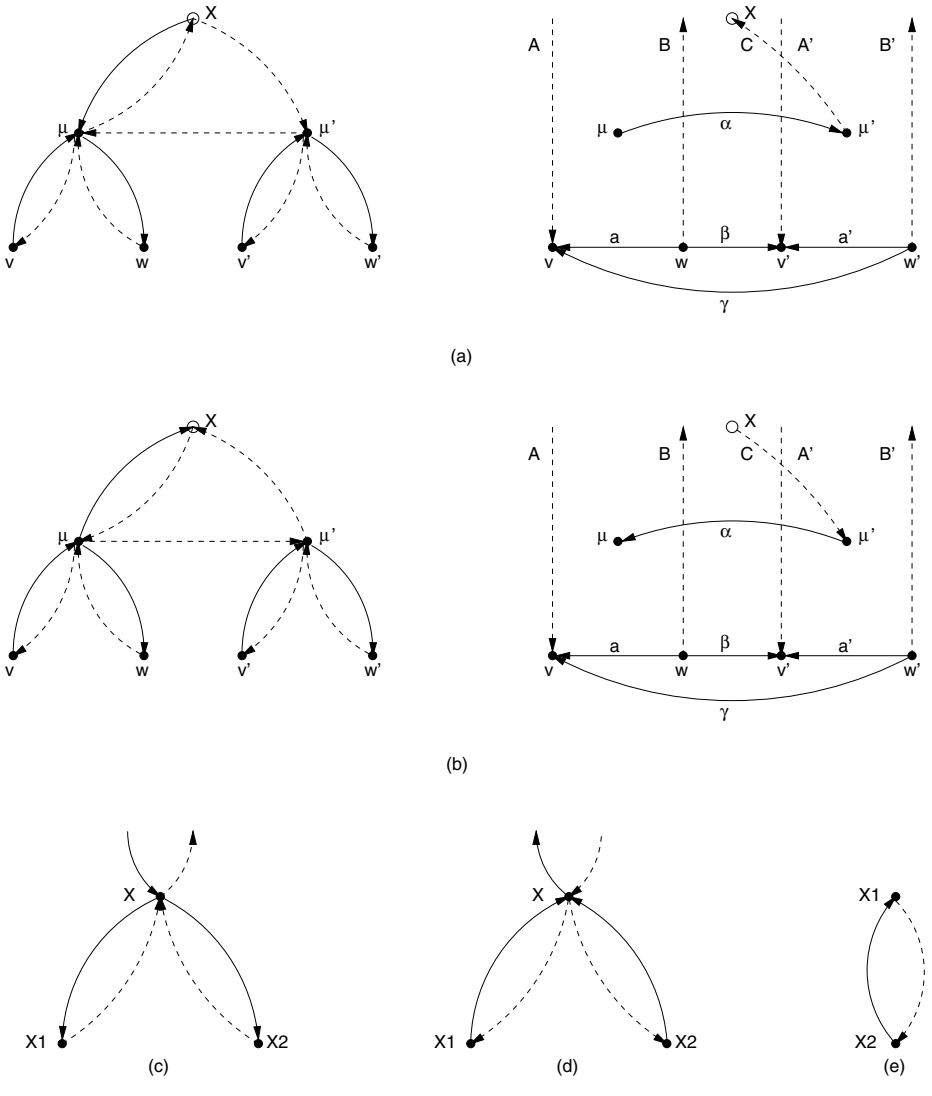


Fig. 3. k-ECSS graph for a generic example. (a) Subgraph for elementary $\mathcal{O}$-module and its parent vertex X. X is drawn hollow as it is in a different module. (b) Subgraph for elementary $\mathcal{I}$-module and its parent. (c) Subgraph for compound $\mathcal{O}$-module with two compound children. The X vertex of the compound module, and the X vertex of the two children, are X, X_1, X_2 respectively. (d) Subgraph for compound $\mathcal{I}$-module with two compound children. (e) Subgraph for the root modules. The X vertex of the two modules are X_1, X_2 respectively. The generic edges for elementary modules are in the right graphs of (a) and (b); generic edges for compound modules and the root modules are in Fig.2(b)–(c). All generic edges have multiplicity 1. Base edges are in the left graphs of (a) and (b), and in (c)–(e). Base edges have multiplicity either k (drawn solid) or $k-1$ (drawn dashed)

The proof begins with a "generic" example – a laminar family $\mathcal{L}$, each set having degree requirement exactly 1, along with a collection of edges such that $(1/3, \ldots, 1/3)$ is the unique solution.

A generic example is constructed by starting with a number of *elementary modules* (Fig.2(a)). A *compound module* is formed by combining two elementary or compound modules (Fig.2(b)). The complete example (Fig.2(c)) is formed by combining two compound modules, each called a *root module*.

Especially important is the asymmetry in the two halves of an elementary module – without it the degree requirements become linearly dependent and we lose uniqueness.

Uniqueness of the solution for $\mathcal{L}$ follows from the following lemma, which indicates how the constraints flow up to the root modules and then back down again. For simplicity we write A instead of x_A, etc.

Lemma 5. *Consider any set of edge weights satisfying all degree requirements of a generic example.*

(i) Any elementary module has $A = B'$ and $B = A'$. Furthermore each of the five internal edges has the same weight as an external edge, A, B or C.

(ii) Any compound module has $A = B'$ and $B = A'$. Furthermore each of the four internal edges has the same weight as an external edge, A, B or C.

(iii) Every edge incident to a root module has weight $1/3$. *Hence every edge of the entire generic example has weight* $1/3$.

Next we show that for any $k \geq 1$ the mincost k-ECSS problem can generate any generic example as the system given by Lemma 1. The graph G for the k-ECSS problem will consist of "base edges" and "generic edges", the latter being the edges of the generic example. The optimum extreme point $\boldsymbol{x}$ will have x_e equal to its upper bound (i.e., its multiplicity in G) for the base edges and equal to $1/3$ for the generic edges.

An elementary module plus vertex X of its parent module (Fig.2(a)–(b)) gives rise to the subgraph of G illustrated in Fig.3(a) or (b). Each compound module (Fig.2(b)) with two compound children gives rise to the subgraph of Fig.3(c)–(d). The two root modules (Fig.2(c)) give rise to the subgraph of Fig.3(e).

In addition to the above vector $\boldsymbol{x}$ we define an integral vector $\boldsymbol{y}$ feasible to (LP), which will be a mincost k-ECSS: y_e equals the upper bound on all base edges and on all A, B, A', B', C' edges; y_e equals 0 on the remaining edges $(a, a', \alpha, \beta, \gamma)$.

5 Mixed Graphs

Any algorithm with approximation ratio r for directed supermodular cut gives an approximation ratio of $\leq 2r$ on mixed graphs if we apply the obvious reduction. Applying iterated rounding directly produces inferior results:

Theorem 3. *For any integer $k > 1$, an infinite family of instances of the minimum cost k-EC augmentation problem on mixed graphs has $(2/n, \ldots, 2/n)$ as*

the unique optimum solution to (LP)*. On these instances round-the-highest has approximation ratio* $n/2$*.*

The example starts with a generic construction as in Section 4.

The theorem omits the case $k = 1$. In fact here iterated rounding does better:

Theorem 4. *For any minimum cost 1-EC augmentation or internal augmentation problem on mixed graphs, any extreme point to* (LP) *has* L_∞*-norm* $\geq 1/4$*. This bound is tight on an infinite family of instances.*

We finally note that when the given graph G is undirected, Jain's algorithm [7] provides a 2-approximation. This is because the given crossing supermodular function can be replaced by a weakly supermodular function.

References

1. Cheriyan, J., Vempala, S.: Edge covers of setpairs and the iterative rounding method. Proc. 8th International Integer Programming and Combinatorial Optimization Conf. (2001) 30–44
2. Cheriyan, J., Vempala, S., Vetta, A.: Approximation algorithms for minimum-cost k-vertex connected subgraphs. Proc. 34th Annual ACM Symp. on Theory of Comput. (2002) 306–312
3. Frank, A.: Kernel systems of directed graphs. Acta Sci. Math., Szeged, Hungary **41** (1979) 63–76
4. Frederickson, G.N., Ja'Ja',J.: Approximation algorithms for several graph augmentation problems. SIAM J. Comput. **10**, 2 (1981) 270–283
5. Fleischer, L., Jain, K., Williamson, D.P.: Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems, submitted; union of An iterative rounding 2-approximation algorithm for the element connectivity problem, Proc. 42nd Annual IEEE Symp. on Foundations of Comp. Sci. (2001) 339–347, and A 2-approximation for minimum cost $\{0, 1, 2\}$ vertex connectivity, Proc. 8th International Integer Programming and Combinatorial Optimization Conf. (2001) 115–129
6. Gabow, H.N.: On the L_∞-norm of extreme points for crossing supermodular directed network LPs. Tech. Rept. CU-CS-991-05, Dept. of Computer Science, University of Colorado at Boulder, 2005.
7. Jain, K.: A factor 2 approximation algorithm for the generalized Steiner network problem. Combinatorica **21** (2001) 39–60
8. Khuller, S., Vishkin, U.: Biconnectivity approximations and graph carvings. J. ACM **41**, 2 (1994) 214–235
9. Kortsarz, G., Nutov, Z.: Approximation algorithm for k-node connected subgraphs via critical graphs. Proc. 36th Annual ACM Symp. on Theory of Comput. (2004) 138–145
10. Melkonian, V., Tardos, E.: Algorithms for a network design problem with crossing supermodular demands. Networks **43**, 4 (2004) 256–265
11. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer-Verlag, NY (2003)
12. Vazirani, V.V: Approximation Algorithms. Springer-Verlag, NY (2001)

Virtual Private Network Design: A Proof of the Tree Routing Conjecture on Ring Networks

C.A.J. Hurkens[1], J.C.M. Keijsper[1], and L. Stougie[1,2]

[1] Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven,
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
{wscor, jkeijspe, leen}@win.tue.nl
[2] CWI, P.O.Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract. A basic question in Virtual Private Network (VPN) design is if the symmetric version of the problem always has an optimal solution which is a tree network. An affirmative answer would imply that the symmetric VPN problem is solvable in polynomial time. We give an affirmative answer in case the communication network within which to create the VPN is a circuit. This seems to be an important step towards an answer to the general question. The proof relies on a dual pair of linear programs and actually implies an even stronger property of VPNs. We show that this property also holds for some other special cases of the problem.

1 Introduction

In this paper, we consider a problem that is known as the *symmetric virtual private network* (VPN) problem, a problem emerging in telecommunication. Think of a large communication network represented by an undirected graph $G = (V, E)$, with a vertex for each user, and an edge for each link in the network. Within this network, a subgroup $W \subseteq V$ of the users, wishes to reserve capacity on the links of the network for communication among themselves: they wish to establish a virtual private network. Vertices in W are also called *Terminals*.

On each link, capacity (bandwidth) has a certain price per unit, $c : E \to \mathbb{R}_+$. The problem is to select one or more communication paths between every pair $\{i, j\}$ of users in W and to reserve enough capacity on the edges of the selected paths to accommodate any possible communication pattern amongst the users in W. Possible communication patterns are defined through an upper bound on the amount to be communicated (transmitted and received) for each node in W, specified by $b : W \to \mathbb{R}_+$. More precisely, a *communication scenario* for the symmetric VPN problem can be defined as a symmetric matrix $D = (d_{ij})_{\{i,j\} \subseteq W}$ with zeros on the diagonal, specifying for each unordered pair of distinct nodes $\{i, j\} \subseteq W$ the amount of communication $d_{ij} \geq 0$ between i and j. A communication scenario $D = (d_{ij})_{\{i,j\} \subseteq W}$ is said to be *valid* if $\sum_{j \in W \setminus \{i\}} d_{ij} \leq b(i),\ \forall i \in W$. We denote the collection of valid communication scenarios by $\mathcal{D}$.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 407–421, 2005.

A network consisting of the selected communication paths with enough capacity reserved on the edges to accommodate every valid communication scenario we call a *feasible VPN*. The (symmetric) VPN problem is to find the cheapest feasible VPN.
There are several variants of the problem emerging from additional routing requirements.

SPR *Single Path Routing:* For each pair $\{i,j\} \subseteq W$, exactly one path $P_{ij} \subseteq E$ is to be selected to accommodate all traffic between i and j. The problem is then to choose the paths P_{ij} such as to minimize $\{\sum_{e\in E} c_e x_e \mid x_e \geq \sum_{\{i,j\}:e\in P_{ij}} d_{ij}, \forall e \in E\ \forall D \in \mathcal{D}\}$.

TTR *Terminal Tree Routing:* This is single path routing with the additional restriction that $\cup_{j\in W} P_{ij}$ should form a tree in G for all $i \in W$.

TR *Tree Routing:* This is SPR with the extra restriction that $\cup_{\{i,j\}\subseteq W} P_{ij}$ is a tree in G.

MPR *Multi-Path Routing:* For each pair $\{i,j\} \subseteq W$, and for each possible path between i and j the fraction of communication between i and j to be routed along that path has to be specified.

The following lemma summarizes the rather obvious relations between the optimal solution values of these variants. By $OPT(\text{SPR})$ we denote the cost of an optimal solution for the SPR variant of the VPN problem. Similar notation is used for the other optimal values.

Lemma 1.

$$OPT(\text{MPR}) \leq OPT(\text{SPR}) \leq OPT(\text{TTR}) \leq OPT(\text{TR}).$$

Proof. SPR is the MPR problem with the extra restriction that all fractions must be 0 or 1. The other inequalities are similarly trivial. □

A prominent open question in VPN design is if SPR is polynomially solvable (SPR $\in P$), cf. Italiano et al. [8]. This question would be answered affirmatively if one could prove that $OPT(\text{SPR}) = OPT(\text{TR})$, since Gupta et al. [6] have shown that TR $\in P$ (see also [9]). Gupta et al. [6] show that $OPT(\text{TR}) = OPT(\text{TTR})$ and that $OPT(\text{TR}) \leq 2OPT(\text{MPR})$. Erlebach and Rüegg [3] proved that MPR $\in P$, which also follows from our LP-formulation in Section 2. They also mention that no VPN instance has been found so far for which even $OPT(\text{MPR}) < OPT(\text{TR})$. Indeed, our conjecture is that $OPT(\text{MPR}) = OPT(\text{TR})$, from which SPR $\in P$ would follow. $OPT(\text{SPR}) = OPT(\text{TR})$ was not known to be true for any class of graphs other than trees. It seems to be a crucial step forward to prove it for circuits, which is implied by the main result in this paper.

Theorem 1. *Let $G = (V, E)$ be a circuit. Then $OPT(\text{MPR}) = OPT(\text{TR})$.*

This theorem is proved in Section 3. The proof boils down to showing that the cost of an optimal solution to TR equals the value of an optimal dual solution

in a formulation of MPR as a linear program (LP). The LP for MPR is given in Section 2, where the conjecture $OPT(\text{MPR}) = OPT(\text{TR})$ is restated in terms of this LP.

In Section 4, we proceed to prove our conjecture, $OPT(\text{MPR}) = OPT(\text{TR})$, for some other special cases. We prove it for any graph G and any cost function c, if the communication bound of some terminal is larger than the sum of the bounds of the other terminals. We also prove it for any graph on at most 4 vertices, and for any complete graph if the cost function c is identical to 1. We also prove that the property OPT(TR)=OPT(MPR) is preserved under taking 1-sums of graphs, implying a common generalization of all the aforementioned results.

The model of the VPN problem presented above was proposed for the first time by Fingerhut et al. [4], and later independently by Duffield et al. [2]. They also formulated the asymmetric version of the problem in which for each node there is a distinction between a bound $b^- : W \to \mathbb{R}_+$ for incoming communication and a bound $b^+ : W \to \mathbb{R}_+$ for outgoing communication. Gupta et al. [6] prove that even the TR problem is NP-hard for the VPN problem with asymmetric communication bounds. However, the TR problem is solvable in polynomial time if $b^-(v) = b^+(v)$ for all $v \in W$. Italiano et al. [8] show that this is true already if $\sum_{v \in W} b^-(v) = \sum_{v \in W} b^+(v)$. The polynomial time algorithm for MPR by Erlebach and Rüegg [3] has been derived for the asymmetric problem. Altin et al. [1] present an LP-formulation of the general MPR VPN problem of polynomial size, immediately implying polynomial solvability of this problem. We devised a similar formulation independently, which we report in a full version of the paper. Gupta et al. [6] and [7] study approximation algorithms for NP-hard versions of the VPN problem.

The challenge remains to prove or disprove that SPR is polynomially solvable on any graph.

2 A Linear Programming Formulation

Let $G = (V, E)$ be a graph, $W \subseteq V$ a set of terminals, $b : W \to \mathbb{R}_+$ communication upper bounds, and $c : E \to \mathbb{R}_+$ unit edge costs. Actually, we will regard b as a function on V rather than on W (defining $b(v) = 0$ for $v \notin W$) and simply identify W with the set of vertices $\{v \in V \mid b(v) > 0\}$. Thus, the triple (G, b, c) defines an *instance* of the MPR or SPR or TR problem. We also use the notation b_v for $b(v)$ and c_e for $c(e)$.

The set of all paths between vertices i and j in W is denoted by $\mathcal{P}_{ij}$. Let $\mathcal{P} = \cup_{\{i,j\} \subseteq W} \mathcal{P}_{ij}$. We introduce the variable x_p for each $p \in \mathcal{P}$. In the MPR VPN problem we are to give (fractional) values to x_p, for all $p \in \mathcal{P}$, such that $\sum_{p \in \mathcal{P}_{ij}} x_p = 1$ for all $\{i, j\} \subseteq W$.

Given values for x_p, $p \in \mathcal{P}$, computing capacity to be reserved on the edges, z_e, $e \in E$, is formulated as follows, where $\alpha_p^e = 1$ if edge e is on path p, and 0 otherwise:

$$
\begin{aligned}
z_e = \max & \textstyle\sum_{\{i,j\}\subseteq W} \sum_{p\in\mathcal{P}_{ij}} \alpha_p^e x_p d_{ij}, \\
\text{s.t. } & \textstyle\sum_{j\in W} d_{ij} \le b_i, && \forall\ i \in W, \\
& d_{ij} \ge 0, && \forall\ \{i,j\} \subseteq W,
\end{aligned}
$$

which, by strong duality, is equal to

$$
\begin{aligned}
z_e = \min & \textstyle\sum_{i\in W} b_i y_i^e, \\
\text{s.t. } & y_i^e + y_j^e \ge \textstyle\sum_{p\in\mathcal{P}_{ij}} \alpha_p^e x_p, \forall\ \{i,j\} \subseteq W, \\
& y_i^e \ge 0, && \forall\ i \in W.
\end{aligned}
$$

The MPR problem is to make a feasible choice for the variables x_p such as to minimize total reservation costs $\sum_{e\in E} c_e z_e$, which by the above can be formulated as

$$
\begin{aligned}
\min & \textstyle\sum_{e\in E} c_e \sum_{i\in W} b_i y_i^e, \\
\text{s.t. } & y_i^e + y_j^e - \textstyle\sum_{p\in\mathcal{P}_{ij}} \alpha_p^e x_p \ge 0, \forall\ \{i,j\} \subseteq W,\ \forall\ e \in E, \\
& \textstyle\sum_{p\in\mathcal{P}_{ij}} x_p = 1, && \forall\ \{i,j\} \subseteq W, \\
& y_i^e \ge 0, && \forall\ i \in W,\ \forall\ e \in E, \\
& x_p \ge 0, && \forall\ p \in \mathcal{P}.
\end{aligned} \tag{1}
$$

MPR is the LP-relaxation of the SPR problem, in which all variables x_p are restricted to be 0 or 1. The dual of MPR is given by

$$
\begin{aligned}
\max & \textstyle\sum_{\{i,j\}\subseteq W} \mu_{ij}, \\
\text{s.t. } & \textstyle\sum_{j\in W} \lambda_{ij}^e \le c_e b_i, && \forall\ i \in W,\ \forall\ e \in E, \\
& \mu_{ij} - \textstyle\sum_{e\in E} \alpha_p^e \lambda_{ij}^e \le 0, \forall\ \{i,j\} \subseteq W,\ \forall\ p \in \mathcal{P}_{ij} \\
& \lambda_{ij}^e \ge 0, && \forall\ \{i,j\} \subseteq W,\ \forall\ e \in E.
\end{aligned} \tag{2}
$$

At this point, let us note that the separation problem over the dual polytope can be solved in polynomial time. There is only a polynomial number of constraints of the first type. For the second set of constraints, suppose we are given λ_{ij}^e $\forall\ \{i,j\} \subseteq W,\ \forall\ e \in E$, and μ_{ij} $\forall\ \{i,j\} \subseteq W$. Take for any pair $\{i,j\} \subseteq W$ the constraints for all $p \in \mathcal{P}_{ij}$ together. To check if they are satisfied is a matter of computing a shortest path between i and j in W, where each edge $e \in E$ has weight λ_{ij}^e. There are only polynomially many i,j pairs in W. It follows, using the ellipsoid method (see [5]), that MPR can be solved in polynomial time, which was also proved in [3].

We proceed with investigating some properties of the problem. A *tree solution* for the instance (G, b, c) is a solution to the TR problem, i.e., a tree in G spanning the set of terminals $W = \{v \in V \mid b(v) > 0\}$, together with optimal capacity reservations on the edges of the tree. It is rather straightforward to give an explicit expression for the cost of a tree solution (cf. [6]). Given a tree T and some edge $e \in T$ let L_e and R_e be the two components of $T - e$. Then the capacity required on e is $\min\{b(L_e), b(R_e)\}$, whence the cost of the tree is given by

$$\sum_{e \in T} \min\{b(L_e), b(R_e)\} c(e). \tag{3}$$

In [6] another expression is derived, based on which polynomial time solvability of TR is concluded.

By weak duality and $OPT(\text{MPR}) \leq OPT(\text{TR})$ the following conjecture is equivalent to the conjecture that $OPT(\text{MPR}) = OPT(\text{TR})$.

Conjecture 1. For any instance (G, b, c), the cost of an optimal tree solution equals the value of an optimal solution of the dual problem (2).

In this paper, we show that Conjecture 1 holds in several special cases, most notably in case G is a circuit, and b and c are arbitrary.

We finish this section with some useful properties. We defer the proofs of the following lemmas, which are in fact all rather easy, to the full version of the paper.

Lemma 2. *Let G be a fixed graph. If for any rational vectors b and c the cost of an optimal tree solution to (G, b, c) equals the value of an optimal dual solution, then the same is true for any instance (G, b, c) where b and c are real vectors.* □

Lemma 3. *For any $\beta \in \mathbb{R}_+$, the instance $(G, \beta b, c)$ has a feasible dual solution of value βK if and only if the instance (G, b, c) has a feasible dual solution of value K. Moreover, $(G, \beta b, c)$ has a tree solution of cost βK if and only if (G, b, c) has a tree solution of cost K. A similar statement holds if c is scaled instead of b.* □

The next lemma claims that edges of zero cost may be *contracted* or *decontracted* when proving Conjecture 1. Contraction of $e = \{u', v'\}$, by identifying the two vertices u' and v' with one new vertex w', transforms G into $G/e = (V \setminus \{u', v'\} \cup \{w'\}, E')$, with $E' := \{\{u, v\} \in E \mid \{u, v\} \cap \{u', v'\} = \emptyset\} \cup \{\{w', v\} \mid \{u', v\} \in E, v \neq v'\} \cup \{\{w', v\} \mid \{v', v\} \in E, v \neq u'\}$. By the contraction of e in the instance (G, b, c) we mean the instance (G', b', c'), where $G' = G/e$, $b'(v) = b(v)$ for $v \neq u', v'$, and $b'(w') = b(u') + b(v')$, and moreover $c'(\{u, v\}) = c(\{u, v\})$, if $w' \notin \{u, v\} \in E'$, and $c'(\{w', v\}) = c(\{u', v\})$ or $c(\{v', v\})$, or both values occur, in case parallel edges arise (edges in G' can be identified with those in $E \setminus \{e\}$). We will denote this contraction (G', b', c') by $(G, b, c)/e$.

Lemma 4. *Let $(G = (V, E), b, c)$ be an instance, where $e \in E$ has $c(e) = 0$. Then $(G = (V, E), b, c)$ has a dual solution of value K if and only if the contraction $(G, b, c)/e$ has a dual solution of value K. Moreover, (G, b, c) has an optimal tree solution of cost K if and only if $(G, b, c)/e$ has an optimal tree solution of cost K.* □

The last lemma says that vertices with a communication bound of 0 and degree 2 in an instance $(G = (V, E), b, c)$ can be neglected in proving Conjecture 1. Suppose $v' \in V$ has degree 2 in V, and $e_1 = \{u, v'\}, e_2 = \{v', w\}$ are the two edges incident with v'. Then *shortcutting* v' results in the instance (G', b', c'), where $G' = (V - v', E \setminus \{e_1, e_2\} \cup \{u, w\})$, $b'(v) = b(v), \forall v \in V - v'$, and $c'(\{u, w\}) = c(e_1) + c(e_2)$, $c'(e) = c(e)$, $\forall e \in E \setminus \{e_1, e_2\}$.

Lemma 5. *Let $(G = (V, E), b, c)$ be an instance, and $v \in V$ a vertex of degree 2 in G with $b(v) = 0$. Denote the instance obtained from (G, b, c) by shortcutting v by (G', b', c'). Then (G', b', c') has a dual solution of value K, if and only if (G, b, c) has a dual solution of value K. Moreover, (G', b', c') has a tree solution of cost K, if and only if the same holds for (G, b, c).* □

3 The Circuit

In this section, we prove Conjecture 1 for circuits. We use the lemmas from the previous section to prove that we may restrict ourselves to *even* circuits, i.e., circuits with an even number of vertices, in which each vertex has communication bound 1.

Lemma 6. *If Conjecture 1 holds for every instance (G, b, c) where G is an even circuit and $b \equiv 1$, then it holds for every instance (G, b, c) where G is a circuit and b is arbitrary.*

Proof. Consider a general circuit instance $(G = C_n, b, c)$. By Lemmas 5 and 2 we can concentrate on instances with rational $b(v) > 0, \forall v \in V$. By Lemma 3 we may divide b by $\gcd\{\frac{1}{2}b(v) | v \in V\}$, making each $b(v)$ a positive even integer. Finally, by Lemma 4, we may substitute each path u, v, w by a path $u, v_1, v_2, \ldots, v_N, w$, where $N = b(v)$, setting $b(v_1) = \ldots = b(v_N) = 1$, and $c(\{u, v_1\}) = c(\{u, v\})$, $c(\{v_k, v_{k+1}\}) = 0$, $c(\{v_N, w\}) = c(\{v, w\})$. Thus, we arrive at an even circuit with $b \equiv 1$. □

From now on consider an even circuit $G = C_{2n} = (V, E)$ on which all vertices have communication bound 1. We number the vertices starting at some vertex and following the circuit in counterclockwise direction $0, 1, 2, \ldots, 2n - 1$. All vertex and edge labels are taken modulo $2n$. The edge $\{i-1, i\}$ is denoted by e_i, or by i in case we use it as an index and no confusion with vertices is possible, $i = 0, \ldots, 2n - 1$. E.g., we will write c_k for the unit cost $c(e_k)$ of edge e_k. On even circuits each edge e_k has an *opposite edge* e_{k+n}.

The cost of the tree solution obtained from the circuit by deleting the edge e_k is denoted by $C(e_k; c)$; we explicitly indicate dependence on c as the unit

edge cost function, since we will use other unit cost functions later. Applying (3) yields $C(e_k;c) = \sum_{i=1}^{n-1} i(c_{k-i} + c_{k+i}) + nc_{k+n}$, which, by regrouping of terms, can be written as

$$C(e_k;c) = \sum_{j=k}^{k+n-1} \sum_{i=j+1}^{j+n} c_i = \sum_{j=k}^{k+n-1} H(j;c), \tag{4}$$

where $H(j;c) := \sum_{i=j+1}^{j+n} c_i$ is the so-called *half-sum* for vertex j, i.e., the sum of the unit costs of the n edges on the path starting in j and ending with the edge opposite to e_j. From this expression the following equations are easily derived.

$$C(e_k;c) - C(e_{k+1};c) = H(k;c) - H(k+n;c),\ \forall k = 0,1,\ldots,2n-1; \tag{5}$$

$$H(j;c) - H(j-1;c) = c_{j+n} - c_j,\ \forall j = 0,1,\ldots,2n-1. \tag{6}$$

Now suppose that the tree obtained by deleting edge e_k has minimum cost among all spanning trees of the circuit, in other words $C(e_k;c) = \min_{e\in E} C(e;c)$. Then, using (5), we have

$$H(k;c) - H(k+n;c) = C(e_k;c) - C(e_{k+1};c) \le 0, \text{ and} \tag{7}$$

$$H(k-1;c) - H(k+n-1;c) = C(e_{k-1};c) - C(e_k;c) \ge 0. \tag{8}$$

Subtracting (8) from (7) and applying (6) yields

$$\begin{aligned} 2c_{k+n} - 2c_k &= H(k;c) - H(k+n;c) - H(k-1;c) + H(k+n-1;c) \\ &= 2C(e_k;c) - C(e_{k+1};c) - C(e_{k-1};c) \le 0. \end{aligned} \tag{9}$$

Consequently, if $c_k = 0$ then $c_{k+n} = 0$ and $2C(e_k;c) - C(e_{k+1};c) - C(e_{k-1};c) = 0$. Hence,

$$\begin{aligned} c_k = 0 \wedge C(e_k;c) = \min_{e\in E} C(e;c) \\ \Rightarrow C(e_{k+1};c) = C(e_{k-1};c) = C(e_k;c) = \min_{e\in E} C(e;c), \end{aligned} \tag{10}$$

i.e., if e_k minimizes $C(e;c)$ and $c_k = 0$, then e_{k-1} and e_{k+1} also minimize $C(e;c)$. In the case of an even circuit with $b \equiv 1$, the constraints of the dual linear program (2) reduce to the following. (In a circuit there are only two possible paths between any pair of vertices.)

$$\begin{aligned} &\textstyle\sum_{j\in V\setminus\{i\}} \lambda_{ij}^e \le c_e, \forall\ i \in V,\ \forall\ e \in E, \\ &\mu_{ij} \le \textstyle\sum_{l=i+1}^{j} \lambda_{ij}^{e_l},\ \ \forall\ \{i,j\} \subseteq V, \\ &\mu_{ij} \le \textstyle\sum_{l=j+1}^{i} \lambda_{ij}^{e_l},\ \ \forall\ \{i,j\} \subseteq V, \\ &\lambda_{ij}^e \ge 0, \qquad\qquad \forall\ \{i,j\} \subseteq V,\ \forall\ e \in E. \end{aligned} \tag{11}$$

Given a cost function $c : E \to \mathbb{R}_+$, we call the set of edges with non-zero unit cost the *support* of c, i.e. $\text{supp}(c) = \{e \in E | c_e > 0\}$. The following lemma is crucial to the main result.

Lemma 7. *Let $G = (V, E) = C_{2n}$ be an even circuit, with $b \equiv 1$. Let $F \subset E$, $F \neq \emptyset$. Then there exists $\hat{c} : E \to \mathbb{R}_+$, not identical to 0, with $\mathrm{supp}(\hat{c}) \subseteq F$, such that for all $f \in F$, $C(f; \hat{c}) = \min_{e \in E} C(e; \hat{c}) = K$ for some $K \geq 0$. Moreover, a dual solution $(\hat{\lambda}, \hat{\mu})$ for the problem with cost function $\hat{c}$ exists with objective value K.*

Proof. The proof is by induction on $|F|$. For $|F| = 1$, suppose $F = \{e_k\}$. Setting $\hat{c}_k = 1$ and $\hat{c}_i = 0$, $\forall i \neq k$. gives $\min_{e \in E} C(e; \hat{c}) = C(e_k; \hat{c}) = 0$. A feasible dual solution with objective value 0 is $\hat{\lambda}^e_{ij} = \hat{\mu}_{ij} = 0$, $\forall e \in E$, $\forall \{i, j\} \subseteq V$. For the case $|F| > 1$ we distinguish three cases.

Case 1: There exists a k such that $e_k \in F$ and its opposite $e_{k+n} \in F$. In this case we call F *singular*. See Figure 1.

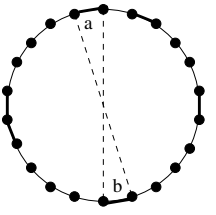


Fig. 1. Example of a singular subset, depicted by bold lines

Consider the cost function $\hat{c} : E \to \mathbb{R}_+$ defined by $\hat{c}_k = \hat{c}_{k+n} = 1$ and $\hat{c}_i = 0$ otherwise. It satisfies $C(e; \hat{c}) = n$, for all $e \in E$. The following dual solution is feasible with respect to this cost function $\hat{c}$ (see (10)) and has objective value $\sum_{i<j} \hat{\mu}_{ij} = n$:

$$\hat{\lambda}^{e_k}_{i,i+n} = \hat{\lambda}^{e_{k+n}}_{i,i+n} = \hat{\mu}_{i,i+n} = 1, \ \forall\, i = 0, 1, \ldots, n-1,$$

$$\hat{\lambda}^e_{ij} = \hat{\mu}_{ij} = 0, \text{ otherwise.}$$

Case 2: F is not singular and there exist k and m, $k < m < k + n$, such that $e_k \in F$, $e_m \in F$, $e_l \notin F$, $\forall k < l < m$, and $e_l \notin F$, $\forall k + n \leq l \leq m + n$. See Figure 2 with $e = e_k$ and $f = e_m$.
We contract the paths $e_k, \ldots, e_m$ and $e_{k+n}, \ldots, e_{m+n}$ to single edges $e' = (k - 1, m)$ and $\bar{e}' = (k - 1 + n, m + n)$, respectively, arriving at a new even cycle (V', E') with $V' = \{m, m + 1, \ldots, k - 1 + n, m + n, \ldots, k - 1\}$ and $E' = \{e_{m+1}, \ldots, e_{k-1+n}, \bar{e}', e_{m+n+1}, \ldots, e_{k-1}, e'\}$. Note that $|V'| = |E'| = 2(n-m-k)$ and that edges that were opposite before contraction remain opposite after contraction. Also, the new edges e' and $\bar{e}'$ are opposite.

Consider the subset of edges $F' = F \backslash \{e_k, e_m\} \cup \{e'\}$. As $0 < |F'| < |F|$ the induction hypothesis tells us that there exists $c' : E' \to \mathbb{R}_+$, not identical to 0, with $\mathrm{supp}(c') \subseteq F'$, a constant $K' = \min_{e \in E'} C(e; c') = C(f; c'), \forall f \in F'$, and a dual solution (λ, μ), with value K'. Since e' is a minimizer of $C(e; c')$, using (9) and the fact that $c'(\bar{e}') = 0$, we have

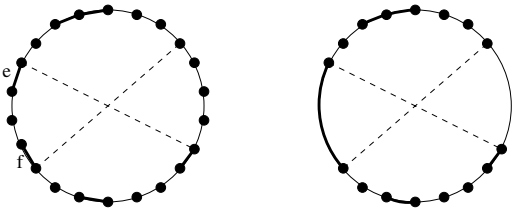


Fig. 2. Contracting the circuit to a smaller one. F depicted by bold lines

$$c'(e') = \tfrac{1}{2}(H'(m+n;c') - H'(m;c')) + \tfrac{1}{2}(H'(k-1;c') - H'(k+n-1;c')),$$

where $H'(j;c')$ is the half-sum for vertex j on the smaller, contracted, circuit with its corresponding cost function c'. Now we define the cost $\hat{c} : E \to \mathbb{R}_+$ from c' as follows:

$$\begin{aligned}
\hat{c}_m &:= \tfrac{1}{2}(H'(m+n;c') - H'(m;c')), \\
\hat{c}_k &:= \tfrac{1}{2}(H'(k-1;c') - H'(k+n-1;c')), \\
\hat{c}_i &:= c'_i, && \forall \{i-1,i\} \in F,\ i \neq k,m, \\
\hat{c}_i &:= 0, && \forall \{i-1,i\} \in E \backslash F.
\end{aligned}$$

Note that $\hat{c}_k + \hat{c}_m = c'(e')$ and

$$\begin{aligned}
\hat{c}_k + H'(k+n-1;c') = \hat{c}_m + H'(m;c') &= \tfrac{1}{2}\textstyle\sum_{e\in E'} c'_e = \tfrac{1}{2}\sum_{f\in F'} c'_f \\
&= \tfrac{1}{2}\textstyle\sum_{f\in F} \hat{c}(f) = \tfrac{1}{2}\sum_{e\in E} \hat{c}(e).
\end{aligned}$$

Hence, for the half-sums in the larger circuit, we have

$$\begin{aligned}
H(j;\hat{c}) &= \hat{c}_m + H'(m;c') = \tfrac{1}{2}\textstyle\sum_{e\in E} \hat{c}(e), && j = k,\ldots,m-1, \\
H(j;\hat{c}) &= \hat{c}_k + H'(k+n-1;c') = \tfrac{1}{2}\textstyle\sum_{e\in E} \hat{c}(e), && j = k+n,\ldots,m-1+n, \\
H(j;\hat{c}) &= H'(j;c'), && \text{otherwise.}
\end{aligned}$$

Using this in (4) yields $C(e_k;\hat{c}) = C(e_m;\hat{c})$, and $C(e_k;\hat{c}) - C(f;\hat{c}) = C(e';c') - C(f;c')$, for all $f \in F'$. Hence, $C(f;\hat{c}) = \min_{e\in E} C(e;\hat{c}) = K, \forall f \in F$, with $K = K' + \frac{1}{2}(m-k)\sum_{e\in E} \hat{c}(e)$.

To define the dual feasible solution with value K we represent V as $V = V' \cup V_{e'} \cup V_{\bar{e}'}$, with $V_{e'} = \{k, k+1, \ldots, m-1\}$ and $V_{\bar{e}'} = \{k+n, k+n+1, \ldots, m+n-1\}$. Again we construct the dual solution $(\hat{\lambda}, \hat{\mu})$ for $(V, E, \hat{c})$ from the dual (λ, μ) associated with (V', E', c'):

$$
\begin{aligned}
\hat{\lambda}^{e}_{ij} &:= \lambda^{e}_{ij}, && \text{for } e \in F \setminus \{e_k, e_m\}, \{i,j\} \subseteq V',\\
\hat{\lambda}^{e_k}_{ij} &:= \tfrac{\lambda^{e'}_{ij}\hat{c}_k}{c'(e')}, && \text{for } \{i,j\} \subseteq V',\\
\hat{\lambda}^{e_m}_{ij} &:= \tfrac{\lambda^{e'}_{ij}\hat{c}_m}{c'(e')}, && \text{for } \{i,j\} \subseteq V',\\
\hat{\lambda}^{e}_{ij} &:= \hat{c}(e), && \text{for } e \in E, i \in V_{e'}, j = i+n,\\
\hat{\lambda}^{e}_{ij} &:= 0, && \text{otherwise},\\
\hat{\mu}_{ij} &:= \mu_{ij}, && \text{for } \{i,j\} \subseteq V',\\
\hat{\mu}_{ij} &:= \tfrac{1}{2}\textstyle\sum_{e\in E}\hat{c}(e), && \text{for } i \in V_{e'}, j = i+n,\\
\hat{\mu}_{ij} &:= 0, && \text{otherwise.}
\end{aligned}
$$

Case 2 is settled by verifying that $(\hat{\lambda}, \hat{\mu})$ satisfies the dual constraints (11), and has value

$$\sum_{i,j\in V, i<j} \hat{\mu}_{ij} = \sum_{i,j\in V', i<j} \mu_{ij} + \sum_{i\in V_{e'}} \hat{\mu}_{i,i+n} = K' + \frac{1}{2}(m-k)\sum_{e\in E}\hat{c}(e) = K.$$

Case 3: F is not singular and between each pair of consecutive edges from F there is exactly one edge that is opposite to some other edge in F. See Figure 3.

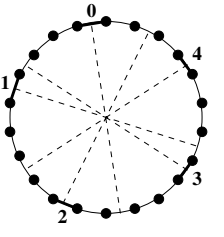


Fig. 3. Example of an edge set with alternating edges and opposites

F consists of an odd number of edges, $2k+1$ say, such that these edges and their opposites are perfectly alternating. As a consequence, an edge $e \in F$ and its opposite $\bar{e} \notin F$, split the other edges of F into two groups of size k. We call this a *uniform* configuration. Let $F = \{f_0, f_1, \ldots, f_{2k}\}$, be numbered counterclockwise, with $f_i = (m_i - 1, m_i)$. Denote the number of nodes between the consecutive

edges f_i and f_{i+1} by ν_i: i.e., $\nu_i := (m_{i+1} - m_i) \bmod 2n$ and $\sum_{i=0}^{2k} \nu_i = 2n$. We note that subscripts i for ν, f etc. are taken modulo $2k+1$. Define

$$\tau_i := \sum_{j=0}^{k} \nu_{i+j} - \sum_{j=k+1}^{2k} \nu_{i+j} = 2(m_{i+k+1} - m_i) \bmod 2n,\ i = 0, 1, \ldots, 2k.$$

By uniformity, $\sum_{j=0}^{k} \nu_{i+j} > \sum_{j=k+1}^{2k} \nu_{i+j}$, whence $\tau_i > 0$, $i = 0, \ldots, 2k$. Moreover, $\frac{1}{2}\tau_i$ equals the number of nodes between $\bar{f}_i$ and f_{i+k+1} (or equivalently between f_i and $\bar{f}_{i+k+1}$).

We propose the following cost function $\hat{c}$. In the definition we use that $\nu_{i+k} = \frac{1}{2}(\tau_i + \tau_{i+k})$.

$$\hat{c}(f_i) := \frac{\nu_{i+k}}{\tau_i \tau_{i+k}} = \tfrac{1}{2}(\frac{1}{\tau_i} + \frac{1}{\tau_{i+k}}),\ i = 0, 1, \ldots, 2k; \quad \hat{c}(e) := 0,\ e \notin F.$$

Observe the following identity, for $i = 0, 1, \ldots, 2k$,

$$\begin{aligned}
\sum_{j=i}^{i+k} \hat{c}(f_j) - \sum_{j=i+k+1}^{i+2k} \hat{c}(f_j) &= \sum_{j=i}^{i+k-1} (\hat{c}(f_j) - \hat{c}(f_{j+k+1})) + \hat{c}(f_{i+k}) \\
&= \tfrac{1}{2} \left(\sum_{j=i}^{i+k-1} \left((\frac{1}{\tau_j} + \frac{1}{\tau_{j+k}}) - (\frac{1}{\tau_{j+k+1}} + \frac{1}{\tau_j}) \right) + (\frac{1}{\tau_{i+k}} + \frac{1}{\tau_{i+2k}}) \right) \\
&= \frac{1}{\tau_{i+k}} > 0.
\end{aligned}$$

As a consequence, we have that consecutive edges from F yield the same cost, as

$$\begin{aligned}
C(f_i; \hat{c}) - C(f_{i+1}; \hat{c}) &= C(f_i; \hat{c}) - C(\bar{f}_{i+k+1}; \hat{c}) + C(\bar{f}_{i+k+1}; \hat{c}) - C(f_{i+1}; \hat{c}) \\
&= \tfrac{1}{2}\tau_i(\hat{c}(f_{i+1}) + \ldots + \hat{c}(f_{i+k}) - \hat{c}(f_{i+k+1}) - \ldots - \hat{c}(f_{i+2k+1})) \\
&\quad + \tfrac{1}{2}\tau_{i+k+1}(\hat{c}(f_{i+1}) + \ldots + \hat{c}(f_{i+k+1}) - \hat{c}(f_{i+k+2}) - \ldots - \hat{c}(f_{i+2k+1})) \\
&= \tfrac{1}{2}\tau_i(-\tfrac{1}{\tau_i}) + \tfrac{1}{2}\tau_{i+k+1}\tfrac{1}{\tau_{i+k+1}} = 0.
\end{aligned}$$

Using (10), we conclude that $C(f_i; \hat{c}) = \min_{e \in E} C(e; \hat{c})$, for all $f_i \in F$.

We now define a feasible dual solution by setting the $\hat{\lambda}$-vector to

$$\hat{\lambda}^e_{st} := \begin{cases} \frac{\tau_i/2}{\nu_i \nu_{i+k}} \hat{c}(e) \left(1 \pm \frac{1}{\tau_i} \frac{1}{\sum_j \hat{c}(f_j)}\right), & \forall e \in E \ \ \forall i = 0, 1, \ldots, 2k \\ & \forall s, t : m_i \le s < m_{i+1}, m_{i+k} \le t < m_{i+k+1}, \\ 0, & \text{otherwise}, \end{cases}$$

$\hat{\mu}_{st} := \sum_{e: s < e \le t} \hat{\lambda}^e_{st},\ \forall \{s, t\} \subseteq V.$

with + in the definition of $\hat{\lambda}^e_{st}$ if $f_i < e \leq f_{i+k}$ and − if $f_{i+k} < e \leq f_{i+2k+1}$. Verifying that this solution satisfies the dual constraints (11) and that its value is $\min_{e\in E} C(e;\hat{c})$ is deferred to the full version of the paper. It is a matter of basic algebraic manipulations but not as easy as verifying feasibility in the previous two cases. □

Theorem 2. *Let $G = (V, E)$ be an even circuit, $c : E \to \mathbb{R}_+$, and $b(i) = 1$, $\forall\, i \in V$. Then the cost of an optimal tree solution equals the value of an optimal dual solution.*

Proof. The proof is by induction on $|\mathrm{supp}(c)|$. The theorem is clearly true if $|\mathrm{supp}(c)| = 1$, when deleting the only edge with positive unit cost yields a tree solution with total cost 0. Setting all dual variables to 0 is feasible and yields value 0.

Now suppose $|\mathrm{supp}(c)| > 1$. Lemma 7 tells us that there exist a non-negative non-zero cost function $\hat{c}$, such that $C(f;\hat{c}) = \min_{e\in E} C(e;\hat{c})$, $\forall f \in \mathrm{supp}(c)$, and a dual solution $(\hat{\lambda}, \hat{\mu})$ w.r.t. $\hat{c}$ with the same objective value. Define cost vector c' as $c' := c - \sigma\hat{c}$, where σ is a scalar chosen such that c' is non-negative and at least one $f \in \mathrm{supp}(c)$ has $c'(f) = 0$. Such a scalar exists, since $\mathrm{supp}(\hat{c}) \subseteq \mathrm{supp}(c)$ and $\mathrm{supp}(\hat{c}) \neq \emptyset$.

Let $C(e^*;c') = \min_{e\in E} C(e;c')$. By (10), we can assume that $e^* \in \mathrm{supp}(c') \subset \mathrm{supp}(c)$. Since $|\mathrm{supp}(c')| < |\mathrm{supp}(c))|$, the induction hypothesis may be applied to c', giving a feasible dual (λ', μ') w.r.t. c' of value $\sum \mu'_{vw} = C(e^*, c')$. The solution $(\lambda, \mu) := (\lambda', \mu') + \sigma(\hat{\lambda}, \hat{\mu})$ is feasible w.r.t. c, as $c = c' + \sigma\hat{c}$. Its value is equal to $C(e^*;c') + \sigma C(e^*;\hat{c}) = \min_{e\in E} C(e;c)$. □

This theorem together with Lemma 6 implies our main result, Theorem 1.

4 Other Cases Where the Conjecture Holds

In this section we present more classes of instances of VPN for which Conjecture 1 holds. We just give short comments on the proofs of the results here, necessarily deferring full proofs to the full version of the paper.

The following lemma provides a useful tool.

Lemma 8. *Let $G = (V, E), b : V \to \mathbb{R}_+, c : E \to \mathbb{R}_+$ be given. Let $H = (V, F)$ be the complete graph on V. Define $c'(\{u, v\})$ for $\{u, v\} \in F$ as the length of a shortest path between u and v with respect to the length function c. If Conjecture 1 is true for the instance (H, b, c') and the optimal tree solution has value K, then it is also true for the instance (G, b, c) with the same optimal tree solution value K.* □

This lemma, the proof of which is rather straightforward, says that we may concentrate on complete graphs and unit costs functions that are metrics, i.e., satisfy the triangle inequality. It is the key to prove Conjecture 1 for a class

of instances, which in itself is not very significant, but the result allows for developing further proof tools presented directly hereafter.

Lemma 9. *For any instance* $(G = (V, E), b, c)$ *such that* $u \in V$ *exists with* $b(u) \geq \sum_{v \neq u} b(v)$*, the cost of an optimal tree solution equals the value of an optimal dual solution.* □

The proof of this lemma is obtained by proposing the shortest path tree, with respect to the length function c on G, from the "dominant" vertex as the optimal tree solution and defining an appropriate dual solution.

The next lemma shows that the property that Conjecture 1 holds is preserved under taking 1-sums. A *1-sum* of two graphs is the graph obtained by identifying a vertex of one graph with a vertex of the other graph. More precisely, let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be disjoint graphs, take any $v_1 \in V_1$ and $v_2 \in V_2$ and identify them, creating a vertex z, which is then the only vertex common to V_1 and V_2, i.e., $V_1 \cap V_2 = \{z\}$. The *1-sum of* G_1 *and* G_2 *in* z is then the graph $G = (V_1 \cup V_2, E_1 \cup E_2)$.

Lemma 10. *Let* $G = (V, E)$ *be the 1-sum of* $G_1 = (V_1, E_1)$ *and* $G_2 = (V_2, E_2)$ *in a vertex* z*. Let* $b : V \to \mathbb{R}_+$*, and* $c : E \to \mathbb{R}_+$ *be given. Then the cost of an optimal tree solution equals the value of an optimal dual solution for the instance* (G, b, c) *if the same holds for every instance of the form* $(G_1, b_1, c|_{E_1})$ *(where* $b_1 : V_1 \to \mathbb{R}_+$ *is arbitrary) and for every instance of the form* $(G_2, b_2, c|_{E_2})$ *(where* $b_2 : V_2 \to \mathbb{R}_+$ *is arbitrary), where* $c|_{E_i}$ *denotes the restriction of* c *to* E_i*,* $i = 1, 2$*.*

The idea behind the proof is to create an optimal tree solution for G_1 with all demand of G_2 concentrated in z, and similarly an optimal tree solution for G_2. The union is a tree solution in G, and an appropriate dual solution is formulated to show optimality of this tree solution.

The next lemma states preservation of the property that Conjecture 1 holds under insertion of edges, with the restriction that the unit cost on an inserted edge is equal to the length of a shortest path between its endpoints. It provides a kind of converse to Lemma 8.

Lemma 11. *Suppose for the instance* $(G = (V, E), b, c)$ *a tree solution of value* K *and a feasible dual solution of value* K *exist, and suppose* $e := \{u, v\} \notin E$ *for* $u, v \in V$*. Then for the instance* (G', b, c') *a tree solution with value* K *and a feasible dual solution of value* K *exist, where* $G' := (V, E \cup \{e\})$*,* $c'|_E = c$*, and* $c'(e)$ *is equal to shortest path distance in* G *between* u *and* v *with respect to the length function* c*.* □

This lemma, the proof of which is rather straightforward, implies that if c is a *tree metric* or a *circuit metric* on G (meaning that it is completely determined by its value on the edges of, respectively, a tree or a circuit in the graph), then Conjecture 1 holds. This follows from the fact that Conjecture 1 is obviously true for trees, and by Theorem 1 for circuits.

Our last proof tool is a way of rephrasing Conjecture 1. Let $\Phi(b, c)$ denote the minimum value of the LP (1) as a function of b and c. For proving Conjecture 1,

by Lemmas 3 and 8, we can concentrate on complete graphs and metrics c scaled such that the optimal tree solution has value 1. Therefore, Conjecture 1 follows from the statement that the minimum over all b and c of $\Phi(b,c)$ subject to the following restrictions is 1:

$$\begin{aligned} &\min \Phi(b,c) \\ &\text{s.t. } b_v \geq 0, && \forall v \in V, \\ &\textstyle\sum_{v \neq s} b_v c_{sv} \geq 1, && \forall s \in V, \\ &c_e \geq 0, && \forall e \in E, \\ &c_{uw} \leq c_{uv} + c_{vw}, && \forall u,v,w \in V. \end{aligned} \tag{12}$$

Here, we use c_{uv} as shorthand for $c_{\{u,v\}}$. The formulation (12), as well as the Lemmas 4, 9, and 11 are used in the proof of the following result. It is a technical proof which involves checking that Conjecture 1 holds, when b is fixed, for vertices c of the polyhedron defined by (12). For graphs on 5 or more vertices, the number and complexity of the cases to check grows enormously, and we have not been able to extend the proof to such graphs.

Theorem 3. *For any instance $(G = (V,E), b, c)$ with $|V| \leq 4$, the cost of an optimal tree solution equals the value of an optimal dual solution.* □

A similar approach, now studying vertices b of (12) for fixed c, is used to prove Conjecture 1 for complete graphs with a cost function which takes the same value on all edges.

Theorem 4. *For any instance (G, b, c) with G a complete graph and $c(e) = 1$, $\forall e \in E$, the cost of an optimal tree solution equals the value of an optimal dual solution.* □

The lemmas and theorems presented above are combined in the following corollary.

Corollary 1. *Suppose $G = (V,E)$ is a graph and $c : E \to \mathbb{R}_+$ is a cost function such that every 2-connected component $H = (V', E')$ of G endowed with the cost function $c|'_E$ is either a circuit, or a graph on at most 4 vertices, or a complete graph with all edge costs equal. Then the cost of an optimal tree solution equals the value of an optimal dual for the instance (G, b, c), for any $b : V \to \mathbb{R}_+$.* □

References

1. A. Altin, E. Amaldi, P. Belotti, and M.Ç. Pinar, *Virtual private network design under traffic uncertainty*, Proceedings of CTW04, 2004, pp. 24–27, extended version at http://www.elet.polimi.it/upload/belotti/.

2. N.G. Duffield, P. Goyal, and A Greenberg, *A flexible model for resource management in virtual private networks*, ACM SIGCOMM Computer Communication Review **29** (1999), no. 4, 95–108.
3. T. Erlebach and M. Rüegg, *Optimal bandwidth reservation in hose-model vpns with multi-path routing*, Proceedings of the 23rd INFOCOM Conference of the IEEE Communications Society, 2004.
4. J.A. Fingerhut, S. Suri, and J.S. Turner, *Designing least-cost nonblocking broadband networks*, Journal of Algorithms **24** (1997), no. 2, 287–309.
5. M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica **1** (1981), no. 2, 169–197.
6. A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, *Provisioning a virtual private network: A network design problem for multicommodity flow*, Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC), 2001, pp. 389–398.
7. A. Gupta, A. Kumar, and T. Roughgarden, *Simpler and better approximation algorithms for network design*, Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC), 2003, pp. 365–372.
8. G. Italiano, S. Leonardi, and G. Oriolo, *Design of networks in the hose model*, Proceedings of the 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks (ARACNE), Carleton Scientific, 2002, pp. 65–76.
9. A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, *Algorithms for provisioning virtual private networks in the hose model*, IEEE/ACM Transactions on Networking **10** (2002), no. 4, 565–578.

Approximation Algorithms for the Minimum Cardinality Two-Connected Spanning Subgraph Problem

Prabhakar Gubbala and Balaji Raghavachari

Computer Science Department, University of Texas at Dallas,
Richardson, TX 75080
{prabha, rbk}@utdallas.edu

Abstract. The minimum cardinality 2-connected spanning subgraph problem is considered. An approximation algorithm with a performance ratio of $9/7 \approx 1.286$ is presented. This improves the previous best ratio of $4/3$ for the problem. An approximation algorithm with a performance ratio of $5/4$ in graphs with a minimum-degree of three at every node is also presented. This class includes all 3-connected graphs.

1 Introduction

We study the minimum-cardinality 2-connected spanning subgraph (2-VCSS) problem. In this paper, connectivity refers to vertex connectivity. When referring to edge-connectivity, we will state so explicitly, and the analogous problem is known as 2-ECSS. A graph is *2-connected* if the deletion of any vertex, with its incident edges, leaves a connected graph. A graph is said to be *2-edge-connected* if the deletion of any single edge does not disconnect it. Finding a minimum-cardinality subset of edges, spanning all the vertices, that is 2-connected (or 2-edge-connected) is NP-hard.

Khuller and Vishkin [12] gave elegant algorithms for 2-VCSS and 2-ECSS with ratios $5/3$ and $3/2$, respectively. Their paper introduced the idea of using depth-first search (DFS) to solve the problem, and all subsequent papers have used a similar framework. Developing on this work further, Garg, Santosh and Singla [7] gave an improved algorithm for 2-VCSS whose performance ratio is $3/2$. Cheriyan, Sebő and Szigeti [1] improved the approximation ratio to $17/12$ for 2-ECSS using an ear decomposition of the graph in which the number of odd-length ears is maximal.

Vempala and Vetta [15] designed new algorithms for both 2-ECSS and 2-VCSS problems, with approximation ratios of $4/3$ for both. The following key concepts introduced by them have paved the way to design better algorithms. They introduced the novel concept of a *beta-structure* to avoid hard configurations. They also preprocessed the graphs to remove cut vertices and adjacent degree two nodes. In addition, they used a solution to the "D2 problem" (minimal subgraph in which each vertex is incident to at least two edges) as the starting

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 422–436, 2005.

point instead of an arbitrary DFS tree. Krysta and Kumar [13] improved the ratio for 2-ECSS to about 1.3326 by employing a smart charging scheme. Jothi, Raghavachari and Varadarajan [10] improved the ratio for 2-ECSS further to 5/4. They also claimed the same ratio for 2-VCSS, but did not provide any details. Later, their claim for 2-VCSS has been withdrawn.

1.1 Our Contributions

We present an improved algorithm, and decrease the best-known approximation ratio from 4/3 [15] to 9/7 for the 2-VCSS problem. We show that even better ratios are possible in graphs in which the minimum degree of each vertex is at least three. We present an algorithm that obtains a ratio of 5/4. There are examples to show that our ratios are tight for the latter case.

2 Definitions

Let $G = (V, E)$ be the given graph, with $|V| = n$. Let *Opt* be an optimal 2-VCSS of G. We will also use *Opt* to denote the cardinality of an optimal 2-VCSS of G, and this should cause no confusion. A node v is called a *beta vertex* if the removal of some two nodes x and y from G (along with their incident edges) results in at least three connected components with one of them containing just v. This definition extends to the concept of a *beta pair*. A subgraph $H = (V, E')$ with $E' \subseteq E$ is called *D2* if the degree of each node is at least 2. A *2-matching* is a subgraph, all of whose nodes have degree 2 or less. It is easily shown that the minimal D2 and the maximal 2-matching problems are closely related, and a solution for one can be easily converted into a solution for the other. Both problems are solved using algorithms for matching.

Vempala and Vetta [15] showed that there is no loss of generality in assuming that G has neither beta vertices or pairs, nor adjacent degree-2 nodes. We will use their preprocessing steps and apply our algorithm only on such preprocessed graphs. In addition, as in [10], we will start with a minimal 3-cycle-free D2, or equivalently, a maximal 3-cycle-free 2-matching. It is easy to convert any minimal D2 into a maximal 2-matching by deleting all but two edges at every node whose degree is more than 2, and a similar transformation in the other direction is also possible.

A set of vertices W is defined to be a *witness* if its removal with its incident edges from the graph results in more components than $|W|$. In any 2-VCSS, there must be at least two edges that connect each component to vertices outside it. Therefore if there are x components when W is removed, there are at least $2x$ edges incident to W in any 2-VCSS. Since $x > |W|$, the sum of the degrees of nodes in W is at least $2|W|+2$. We will show that $|V|$ plus the number of disjoint witness sets in the graph is a lower bound on a 2-VCSS.

A pair of vertices is called a *nontrivial 2-cut* if its removal from the graph results in at least two components that are not singleton vertices. We will show that the 2-VCSS problem in graphs with nontrivial 2-cuts can be solved using

divide-and-conquer, by splitting the graph into two parts, finding 2-connected subgraphs for each and then combining the solutions to generate a 2-VCSS for the given graph.

3 New Techniques and Lower Bounds: How to Break the 4/3 Barrier

3.1 The Degree Lower Bound

We defined a 'witness' earlier, as a set of vertices whose removal generates more components than its cardinality. Vempala and Vetta [15] used the D2 lower bound, which is the minimum number of edges in a subgraph (not necessarily connected) in which each vertex has degree greater than or equal to 2. We show an example in which none of the previous methods can guarantee a ratio better than 4/3. We show that a collection of disjoint witness sets gives a new lower bound on an optimal 2-VCSS, and use it to improve the approximation ratio to $9/7 \approx 1.286$. Let the number of paths in a maximum cardinality 2-matching of G be denoted by p. Jothi, Raghavachari and Varadarajan [10] proved that $Opt \geq n + p$. If the number of disjoint witness sets in the graph is w, we show that $Opt \geq n + w$. We can combine the two lower bounds and write it as $Opt \geq n + max\{w, p\}$. We have examples where this new lower bound allows us to prove better ratios than just the D2 lower bound. Using this new lower bound, we are able to tackle some of the tough configurations. The graph in Fig. 1 demonstrates the use of a witness. Previous algorithms for 2-VCSS do not use the witness lower bound, and therefore would get only a 4/3 ratio for this example.

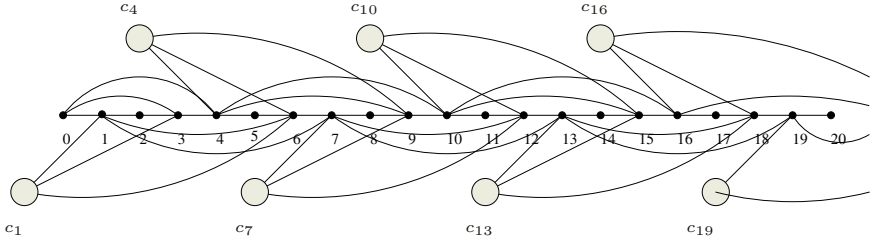


Fig. 1. This example demonstrates the use of improved lower bound. The set $\{1, 3, 4, 6\}$ forms a witness since the deletion of vertices in this set results in five components: i.e., $\{0\}, \{2\}, \{5\}, \{c_1\}$ and $\{7 \ldots\}$

Proposition 1. *Let $G = (V, E)$ be a graph defined on $n \geq 4$ vertices. Let w be the number of disjoint witness sets in a graph and Opt be the cardinality of an optimal 2-VCSS of G, then $Opt \geq n + w$.*

Proof. Every vertex has a degree greater than or equal to 2 in any 2-VCSS of G, including an optimal solution. This can be viewed as the "degree lower bound"

on *Opt*. In addition, each witness W contributes an extra 2 to the degree lower bound because the removal of W generates more components than its cardinality. Since each of these components have at least 2 edges connecting them to W in any 2-VCSS, there are at least $2|W| + 2$ edges incident to W in it. Since the witness sets are disjoint, the lower bounds on their degrees are cumulative. Therefore the sum of the degrees of all vertices in *Opt* is at least $2(n + w)$, and $Opt \geq n + w$.

3.2 Charging Strategy

Jothi, Raghavachari and Varadarajan [10] proved that $Opt \geq n + p$, where p is the number of paths in maximal 3-cycle free 2-matching. If the number of disjoint witness sets in the graph is w, we showed that $Opt \geq n + max\{w, p\}$. Therefore, if our target ratio is $\alpha \geq 1$, we can use up to $\alpha(n+max\{w, p\})$ edges. We account for the edges in our solution by distributing "charges" to different objects (such as vertices, paths and witness sets), such that the total charge is at most $\alpha(n + max\{w, p\})$. In each step of the algorithm, as edges are added to our solution, we pay for the edges using the charges available to the objects involved. Each vertex is allocated a "vertex charge" of α. Each path in the initial 2-matching is assigned a "path charge" of $(2\alpha - 2)$, and each witness is assigned $(2 - \alpha)$ as "witness charge".

Proposition 2. *If the goal is to obtain an α-approximation algorithm, we can always assign $(2\alpha - 2)$ charge to each witness and $(2 - \alpha)$ charge to each path.*

Proof. We know that $Opt \geq n+p$, where p is the number of paths in an optimal (maximum) 2-matching. Also $Opt \geq n + w$, where w is the number of disjoint witness sets. So, $opt \geq n + max\{w, p\}$. If each witness is assigned $(2\alpha - 2)$ charge, total charge assigned to all witnesses in the graph is $(2\alpha - 2)w$. If each path is assigned $(2 - \alpha)$ charge, total charge assigned to all paths in the graph is $(2 - \alpha)p$. So, the total charge assigned to both paths and witnesses is $(2\alpha - 2)w + (2 - \alpha)p \leq (2\alpha - 2)max\{w, p\} + (2 - \alpha)max\{w, p\} = \alpha\ max\{w, p\}$. Since $opt \geq n+max\{w, p\}$, we can distribute $\alpha\ max\{w, p\}$ to the paths and witnesses as stated in this proposition, when our goal is to obtain an α-approximation algorithm.

3.3 Decomposition Techniques

In this subsection, we present decomposition techniques that allow us to focus on graphs with a simplified structure. We use this decomposition procedure after the graph has been preprocessed to remove beta vertices, beta pairs and adjacent degree-2 vertices [15].

Nontrivial 2-Cuts. We defined a pair of vertices to be a nontrivial 2-cut if when removed from the graph, the remaining graph has at least two components that are not singleton vertices (see Fig. 3). We show that if the input graph has a nontrivial 2-cut, it can be partitioned into 2 graphs, and the 2-VCSS problem can be solved independently in these two graphs. If any of these graphs has less

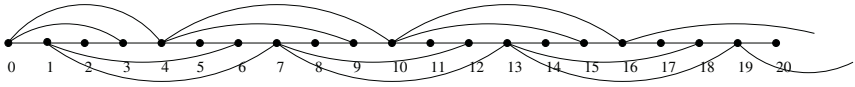


Fig. 2. This example demonstrates the use of a nontrivial 2-cut. There are several nontrivial 2-cuts in this example. { 4, 7} is one of them. This is one of the tough configurations. We would preprocess the graph for nontrivial 2-cuts and apply our algorithm on decomposed graphs as explained below. So, our algorithm does not have to deal with this configuration

than 10 nodes, the 2-VCSS problem on it is solved optimally. Otherwise, we show that an approximation algorithm that outputs a solution that uses at most $\alpha Opt - 4(\alpha - 1)$ edges for graphs without nontrivial 2-cuts can be used to find an $\alpha Opt - 4(\alpha - 1)$ approximation algorithm for arbitrary graphs. We decompose the given graph using the procedure described below until each graph has no nontrivial 2-cuts. Then we find approximate solutions for each of these graphs, and then combine the solutions to generate a solution to the original graph. The graph in Fig. 2 demonstrates the use of 2-cut. On this example, the previous algorithms find only a 4/3 approximation. Since it has a nontrivial 2-cut, our algorithm decomposes the graph into two parts and solves them separately.

We assume now that we have an approximation algorithm for 2-VCSS in graphs with out non trivial 2-cuts that finds a solution containing at most $\alpha Opt - 4(\alpha - 1)$ edges. Suppose we are given a graph that has nontrivial 2-cuts. We find a 2-VCSS in it using the following recursive algorithm. Let C_1 and C_2 be two components, that are not singleton vertices, resulting from the given graph $G = (V, E)$ when a nontrivial 2-cut $\{u, v\}$ is deleted. Let G_1 be the graph that

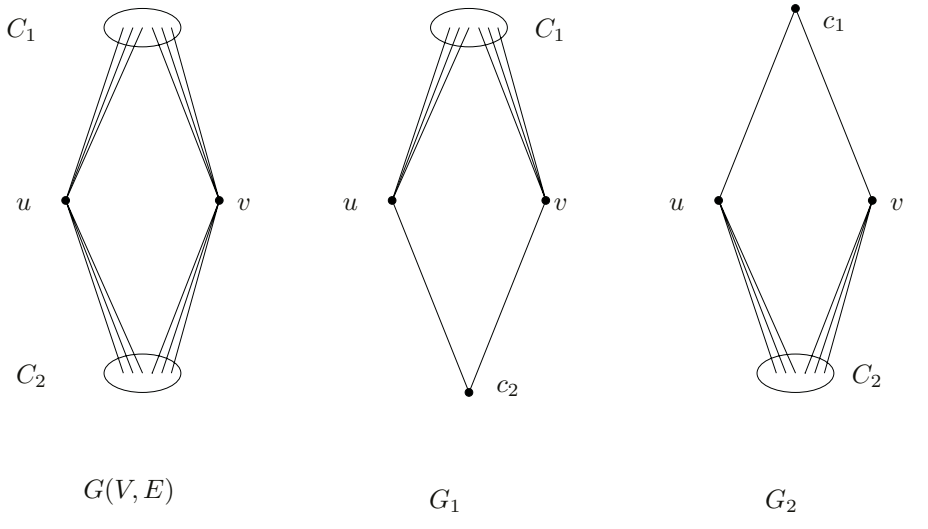


Fig. 3. { , } is a nontrivial 2-cut in this picture. C_1 and C_2 are components that are not singleton nodes. G_1 and G_2 are also shown

results from G after contracting C_2 in to a single node, and let G_2 be the graph resulting from G after contracting C_1 in to a single node (see Fig. 3). We now find an approximate 2-VCSS in each of G_1 and G_2 recursively, and combine the solutions, dropping the edges incident to the contracted nodes (see Fig. 3). We will describe an algorithm later that finds an approximate 2-VCSS of G with at most $\alpha Opt - 4(\alpha - 1)$ edges if G has no nontrivial 2-cuts. We first show in the following proposition that the divide-and-conquer scheme above combines approximate solutions to G_1 and G_2 to get an approximate 2-VCSS of G with at most $\alpha Opt - 4(\alpha - 1)$ edges.

Proposition 3. *Consider a graph $G = (V, E)$ with $|V| = n \geq 4$ vertices. Let C_1 and C_2 be some components that are not singleton vertices in G when vertices $\{u, v\}$ are deleted with their incident edges. Let G_1 be the graph resulting from G after C_2 is contracted to a single node, and let G_2 be the graph G that results after contracting C_2 to a single node (see Fig. 3). If both G_1 and G_2 have more than 9 nodes in each of them, we can combine $\alpha Opt - 4(\alpha - 1)$ approximate 2-VCSS solutions to G_1 and G_2 to obtain an $\alpha Opt - 4(\alpha - 1)$ approximate 2-VCSS solution of G. If G_1 has less than 10 nodes, we can combine optimal 2-VCSS solution of G_1 and $\alpha Opt - 4(\alpha - 1)$ approximate 2-VCSS solution of G_2 to obtain an $\alpha Opt - 4(\alpha - 1)$ approximate 2-VCSS solution of G.*

Proof. Let the contracted node in G_1 be c_2 and the contracted node in G_2 be c_1. Consider an optimal 2-VCSS of G. Let the number of its edges contained entirely within $C_1 \cup \{u, v\}$ be Opt_1, and the number of its edges contained within $C_2 \cup \{u, v\}$ be Opt_2. Observe that $Opt = Opt_1 + Opt_2$, since an optimal solution will not have an edge between u and v (because it would not be critical to its 2-connectivity).

Let $OPT(G_1)$ and $OPT(G_2)$ be the cardinalities of optimal 2-VCSS solutions of G_1 and G_2 respectively. By construction, c_1 and c_2 are degree-2 nodes in G_2 and G_1 respectively. Observe that $OPT(G_1) = Opt_1 + 2$ and $OPT(G_2) = Opt_2 + 2$ as the combination of 2-VCSS solutions of G_1 and G_2 with out the nodes c_1 and c_2 and the edges on them gives the 2-VCSS solution for G and any 2-VCSS solution for G can be decomposed in the solutions for G_1 and G_2 by contraction C_2 and C_1 respectively.

Consider the case when G_1 and G_2 have more than 9 nodes in each of them. We are given $\alpha Opt - 4(\alpha - 1)$ approximate 2-VCSS solutions to G_1 and G_2. We delete c_2 and two edges incident on it from G_1's solution and c_1 and two edges incident on it from G_2's solution and combine it to get a solution for G. So, solution for graph G would incur $\alpha OPT(G_1) - 4(\alpha - 1) + \alpha OPT(G_2) - 4(\alpha - 1) - 4$ edges. i.e., $\alpha(Opt_1 + 2) - 4(\alpha - 1) + \alpha(Opt_2 + 2) - 4(\alpha - 1) - 4$ This is equal to $\alpha Opt - 4(\alpha - 1)$.

Consider the case when G_1 has less than 10 nodes. In this case, we find an optimal 2-VCSS of G_1, and a 2-VCSS of G_2 that uses at most $\alpha Opt - 4(\alpha - 1)$ edges, We delete c_2 and two edges incident on it from G_1's solution, and also delete c_1 and two edges incident on it from G_2's solution, and take the union of the two solutions and return it as a solution for G. So, solution for graph G would incur $OPT(G_1) + \alpha OPT(G_2) - 4(\alpha - 1) - 4$ edges. i.e., $(Opt_1 + 2) +$

$\alpha(Opt_2 + 2) - 4(\alpha - 1) - 4$. This is at most $\alpha Opt - (\alpha - 1)(2 + Opt_1)$. We can see that $(\alpha - 1)(2 + Opt_1) \geq 4(\alpha - 1)$ as $Opt_1 \geq 2$ (since components are not singleton nodes).

3.4 Plunder

The operation in which we replace an edge with a path in an ear is plunder. It has an effect of adding at least one edge to the ear. The graph in Fig. 4 demonstrates the use of plunder operation. Nodes v_0 and v_1 together represent the current 2-connected component, with v_1 as hinging node for the remainder of the path. Node c_1 is a child of v_1 that heads a path. Here, the graph is Hamiltonian, and the 2-matching consists of all 6-cycles. The previous algorithms find only a 4/3 approximation, because they can add only 3 new nodes for each ear. We use plunder operation and add at least 7 new nodes with 2 ears. The two ears we choose are $v_1v_7v_6v_5v_4v_3v_0$ and $v_1v_2v_3$. But with there ears, we added only 6 new nodes. Now we use plunder operation to replace the edge (v_1, v_7) with a path $v_1c_1 \ldots v_7$. So, we added at least 7 new nodes with two ears. Plunder seems to be necessary to get a 9/7 approximation in this example.

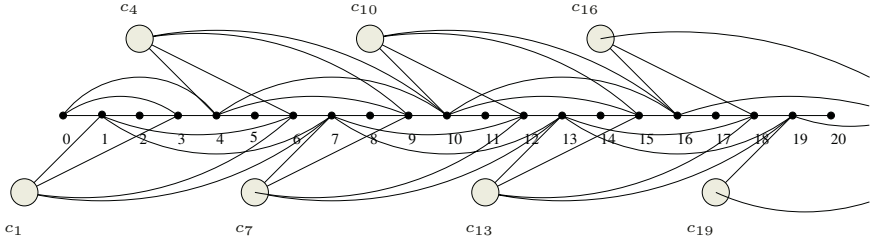


Fig. 4. This example demonstrates the use of the plunder operation. Here, the graph is Hamiltonian, and the 2-matching consists of all 6-cycles. Plunder operation helps us to get 9/7 approximation in this example. The previous algorithms find only a 4/3 approximation, because they can add only 3 new nodes for each ear. The plunder operation seems to be necessary to get a 9/7 approximation in this example

3.5 Expendable Edge

An expendable edge is an edge that is not included in some optimal 2-VCSS of the graph. In general, identifying whether an edge is expendable is NP-hard. This can be argued as follows. If we have a polynomial time subroutine to find all the expendable edges, we can use that as a subroutine to find an optimal solution by finding all the expendable edges in the given graph and deleting them. When there are no expendable edges. we are left with an optimal solution.

However, there are instances when we can identify that some edge in the graph is expendable using a polynomial time algorithm, and in these cases, we can safely delete that edge from the graph and run the 2-VCSS algorithm on the remaining graph. This will not affect the performance of the algorithm since

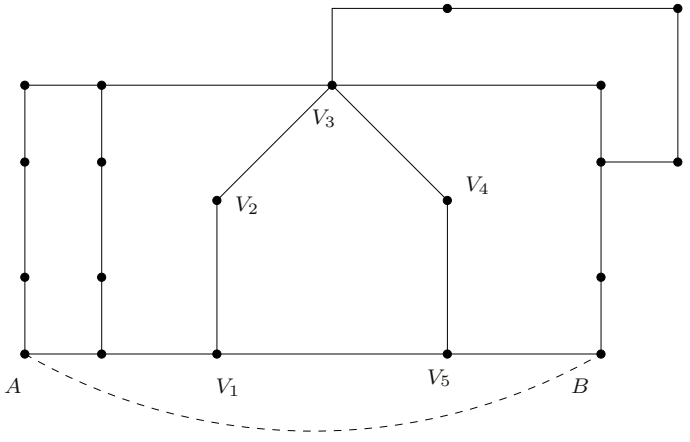


Fig. 5. Example that shows an expendable edge. In this example, if edge(A, B) exists, ($_1$, $_5$) is an expendable edge. Otherwise, $_2$ is a beta vertex. i.e, removal of $_3$ and $_1$ results in three components and one of them is just $_2$

deleting an expendable edge does not change the cardinality of *Opt*. Deleting expendable edges helps in avoiding some tough configurations. We define a specific configuration with an expendable edge and prove that it is not there in at least one optimum solution in the following proposition. We delete all such expendable edges in the given graph before applying the our algorithm to the graph. So, our algorithm assumes there are no expendable edges in the graph as in Proposition 4. The graph in Fig. 5 has an expendable edge (V_1, V_5) if edge (A, B) exists. Otherwise, v_2 is a beta vertex, i.e, removal of v_3 and v_1 results in three components and one of them is just v_2.

Proposition 4. *Consider a graph $G = (V, E)$ with 6 or more nodes without beta vertices, beta pairs and 2-cuts. Let $v_1v_2v_3v_4v_5$ be a path in the graph. Let v_2 and v_4 be degree-2 vertices. Then, the edge (v_1, v_5) is an expendable edge, i.e., there exists an optimal 2-VCSS that does not include (v_1, v_5).*

Proof. We will prove this lemma by contradiction. Suppose all optimal 2-VCSS solutions have the edge (v_1, v_5). Let $G' = (V, E')$ be one such optimal solution. Since v_2 and v_4 are degree-2 vertices, all their incident edges must be in E'. With the addition of the edge (v_1, v_5) to these edges, there are two vertex disjoint paths between v_1 and v_5 in G': v_1v_5 and $v_1v_2v_3v_4v_5$. Therefore there are no other vertex-disjoint paths between v_1 and v_5 in G', because otherwise edge (v_1, v_5) will not be critical in G' for 2-connectivity, contradicting the assumption that it is an optimal 2-VCSS.

Let S be a set of all vertices except v_3 and v_2 that have to go through either v_1 or v_3 to reach v_5 in G'. Let D be the set of all vertices except v_3 and v_4 that have to go through either v_3 or v_5 to reach v_1 in G'. Then, v_1 is a member of set S and v_5 is a member of set D. If edge (u, v) exists such that $u \in S$ and $v \in D$ and $(u, v) \neq (v_1, v_5)$ in graph G, we can add that edge and delete (v_1, v_5)

in G'. The resulting graph is also an optimal 2-VCSS, which does not include (v_1, v_5). This contradicts the assumption that (v_1, v_5) is in all optimal solutions. Therefore, G has no edges connecting a vertex in S with a vertex in D other than (v_1, v_5).

If there is no such edge in graph G, there are three cases to consider. The first case is when both S and D include nodes other than v_1 and v_5, respectively. Removal of v_1 and v_3 generates three components: $\{v_2\}$, S, and $D \cup \{v_4\}$, which makes v_2 a beta vertex. This is a contradiction, because G has no beta vertices.

The second case to consider is S has only one member in it, v_1. Removal of v_3 and v_5 generates three components: $\{v_4\}$ $\{v_2, v_1\}$ and D, which makes v_4 a beta vertex. This is also a contradiction. The third case is when D has only one member in it, v_5. This case is dealt similar to the second case as it is symmetric. Hence the claim is proved.

4 Overview of the Algorithm

We start with the same preprocessing steps used by Jothi et al. [10]. In addition, we look for 2-cuts, expendable edges and decompose the graph as described in the previous section. We will assume that the graph G has already been preprocessed to ensure that it has no beta nodes and pairs, 2-cuts, expendable edges (as in Proposition 4), and adjacent degree-2 nodes.

Algorithm VertexConnect(G)

1. We use Hartvigsen's algorithm [8] to find, in polynomial time, a maximal 3-cycle-free 2-matching (maximal subgraph without 3-cycles, in which each vertex is incident to at most two edges). Let the 2-matching that we obtain be a collection of cycles C and a set of paths P.
2. BuildDFS; We elaborate this procedure in the next section.
3. For each DFS path D_i, twoConnectDfsPath(D_i); We elaborate this procedure later.

Some useful propositions are stated here. Procedures BuildDFS and twoConnectDfsPath are elaborated later.

Proposition 5. *Let $G = (V, E)$ be a 2-connected graph defined on $n \geq 4$ vertices. An optimal 2-VCSS of G has no 3-cycles.*

Proposition 6. *Let C and P be a set of cycles and paths in a maximal 3-cycle-free 2-matching. There is no loss of generality in assuming that for each path $A \in P$, each end vertex of A has a neighbor in G outside A.*

Proof. If an end vertex u of A is connected only to nodes of A, then A can be split into a cycle and a smaller path using an edge from u to the farthest vertex in A to form a cycle. This operation yields another 2-matching with the same number of edges. If the split results in a 3-cycle, then u is a degree-2 node with

neighbors v and w. By Proposition 5, edge (v, w) is not in Opt and therefore we delete it from G, making u and internal node of A. Repeating the above steps finally yields a maximal 3-cycle-free 2-matching that satisfies the proposition.

4.1 BuildDFS Procedure

With C and P (cycles and paths in a maximum 2-matching) on hand, we build a DFS tree of G using the following strategy. When the first node x_1 of a cycle $X \in C$, where $X = \{x_1, x_2, \ldots, x_k\}$, is discovered, we will make the DFS go through the nodes of the cycle in sequence until it reaches the end of the cycle. Upon reaching a leaf node, we take the subgraph induced by the last 9 nodes and see if there is a different way of traversing them such that the new leaf vertex is adjacent to an unvisited vertex. If such a reorganization is possible, we do so, and the DFS is able to continue further. As a consequence, if x_k is a leaf node in the DFS tree, then all neighbors of x_2 and x_k in G are ancestors of x_k. In other words, these nodes do not have neighbors in G that are unexplored when X is discovered by DFS.

When a path $Y \in P$ is encountered by DFS, it is not necessary that the initial node encountered be an end vertex of the path. We use the following idea from [13] to handle paths. If DFS enters Y at one of its end vertices, then it goes through the vertices of the path in that sequence. On the other hand, if a node in the middle of Y is encountered, we continue along the longer segment of Y. We have some exceptions to this rule like when we hit a 4-path in the middle, we do not take the longer route. Instead, we take two nodes with the DFS path and leave 2 nodes. Also, we do not further break a 3-path in to 2-path and a single node path. Instead, we 2-connect it using the vertex charges and the path charge it has been allocated. We could not provide all the details here due to space restrictions. The proof of the following proposition involves describing all these exceptions. We allocate the path charge of $(2 - \alpha)$ received by paths in P to their smaller segments as follows:

Proposition 7. *If $\alpha \geq 5/4$, the path charge of $(2 - \alpha)$ available to each path can be assigned as follows: 3-node segments of a path receive at least $2 - \alpha$ each, 2-node segments receive at least $3 - 2\alpha$ each, and 1-node segments receive at least $2 - \alpha$ each.*

We break the DFS tree into a sequence of paths in a natural way. The first path starts at the root and goes down to the first leaf encountered. Every time the DFS finds an unvisited vertex and starts a new search, we start building a new path that extends up to the first leaf node encountered by it. Let these DFS paths be $D_1, D_2, \ldots, D_p$.

4.2 Two-Connecting the DFS Paths

We first start by 2-connecting D_1, the first path in the DFS tree. We will try to build an open ear decomposition of the nodes of D_1, starting from the leaf.

Once the ear decomposition reaches the root vertex, all vertices of D_1 are two-connected, which we will call as the "core". We then proceed iteratively as follows. Out of the remaining paths, we select the path D whose head is visited earliest by DFS and 2-connect their nodes to the core using the same ideas. Since there are no cross edges with respect to a DFS tree, we are able to 2-connect them one path at a time. At the end of the sequence, we get a solution that is 2-connected and, in fact, we compute an ear decomposition of the solution in which most ears contain at least 4 new nodes. There may be a few smaller ears generated by residual segments with three or fewer nodes, and we account for their edges by combining vertex charges, path charges and witness charges. Since the paths D are generated by DFS, non-tree edges of G are all back edges.

If the graph under consideration has degree 3 or more at all vertices, and the length of the DFS path is less than 8, we invoke procedure SmallPathsforDegree-3. Otherwise, we 2-connect the DFS path in three steps. i.e., basis step, induction step and final step. In general graphs, if the length of the DFS path is less than 11, we call procedure 2-connectSmallPaths. Otherwise, we 2-connect the DFS path in three steps as before: basis step, induction step and final step.

Algorithm twoConnectDfsPath(D_i)

1. If the given graph has degree three or more at all nodes and $|D_i| \leq 7$, call procedure SmallPathsforDegree-3 and return.
2. Otherwise if $|D_i| \leq 10$, call 2-connectSmallPaths and return.
3. Basis step: find a cycle, starting from the leaf node.
4. Induction step: extend the open ear decomposition of D_i by adding an ear that consumes additional nodes of the path.
5. Finishing step: connect the ear decomposition of D_i to the core, without generating any cut vertices.

4.3 Two Connecting Small Paths

We will show that we can 2-connect each small path using the vertex charge of α received by each vertex, path charge it may have been allocated and the witness charge it may have been allocated. Paths with fewer than 4 nodes do not have enough vertex charge available to pay for the edges needed to 2-connect them. Since each cycle in C has at least four edges, a path D_i with three or fewer nodes must be a segment of a path in P. So they have path charge on them. We 2-connect these paths and pay for the edges by combining their vertex charge and path charge. Two connecting small paths with more than or equal to 4 nodes is explained in the following paragraphs.

Degree-3 Graphs. If $4 \leq |D_i| \leq 7$, a single open ear that extends from the core through all the vertices of D_i is constructed. Edges of the ear are paid by vertex charges, path charges that may have been allocated and witness charges that may have been allocated. Details are given in full paper.

General Graphs

1. If $4 \leq |D_i| \leq 6$, a single open ear that extends from the core through all the vertices of D_i is constructed. Edges of the ear are paid by vertex charges, path charges that may have been allocated and witness charges that may have been allocated. Details are given in full paper.
2. If $7 \leq |D_i| \leq 10$, D_i is 2-connected to the core using two open ears. Edges of the ears are paid by vertex charges, path charges that may have been allocated and witness charges that may have been allocated. Details are given in full paper.

4.4 Basis Step

In general, our strategy is to try to construct long cycles starting at a leaf. If we find a long enough cycle starting from the leaf, we will generate enough surplus charge to pay for the finishing step.

Degree-3 Graphs. In this case, we prove a ratio of $\alpha = 5/4$. Therefore, each vertex has a charge of $5/4$ at each vertex, and a charge of $1/2$ is available for each witness. We show that when the path is long enough, either we can find a 8-cycle, or we can find a 6 or 7-cycle with a witness. If we found an 8-cycle, the nodes have a total charge of 10 available, and we pay for the 8-cycle, and we have an excess charge of 2. On the other hand, we find a witness which has an extra charge of $1/2$ available, and including it, there is an extra charge of 2. Therefore, we are able to generate a surplus charge of 2, and this will be utilized by the finishing step.

General Graphs. Here, our target ratio is $\alpha = 9/7$. Therefore each vertex has a charge of $9/7$, and a charge of $4/7$ is available to each witness. In the basis step, if the path is long enough, we show that we can always find one of the following: (i) 7-cycle, or (ii) 6-cycle with a witness or (iii) 2 ears to two connect first 11 nodes. Outline of the basis step is the following. We prove that we can always find a 5 or longer cycle in general graphs. Then, we can either extend the 5-cycle to 6 or longer cycle or find an expendable edge. If we find an expendable edge, we delete it and restart the algorithm. Otherwise, we find a 6-cycle and show that we can either extend it to a 7 or longer cycle or find two ears to two connect first 11 nodes starting from the leaf. In all cases, it can be verified that we generate a surplus charge of 2, which is sufficient to pay for the finishing step.

4.5 Induction Step

After finding a first long cycle starting at the leaf, we assume that a 2-connected component is hanging at the bottom of a DFS path. Now we try to find an open ear that has two end points in the component and that includes several contiguous new nodes which can pay for the edges of the ear. This step is called induction step. With each step, the size of the 2-connected component hanging at the bottom increases and the number of nodes needed to be 2-connected in a

DFS path decreases. We continue performing induction step until the final step in which we 2-connect the DFS path to the core. We will discuss the final step in the next section.

If every node in the graph has degree three or more, we prove a lemma that guarantees that either an open ear can be added that includes 4 or more new nodes, or two open ears that include 8 or more new nodes. This is called as the induction step for degree-3 graphs. For general graphs, we prove a lemma that guarantees that either an open ear can be added that includes 4 or more new nodes, or two open ears that include 7 or more new nodes. This is called induction step for general graphs. We mention the induction lemma's here. Proof's for these lemma's are provided in the full paper.

Lemma 1 (Induction Step for Degree-3 Graphs). *Consider a graph in which each vertex has degree three or more. Let D be a DFS path whose vertices are labeled $v_0, v_1, v_2, v_3, ..., v_q$ starting from the leaf vertex. Nodes v_0 and v_1 together represent the current 2-connected component, with v_1 as the hinging node for the remainder of the path. If there is a DFS path headed by v_1's child, let the DFS path be D_1. It is possible to do one of the following:*

1. *Find an ear with two end vertices v_0 and v_1 that includes all vertices in $\{v_0, v_1, v_2, \ldots, v_k\}$ for some $k \geq \min(q, 5)$.*
2. *If D_1 has four vertices, (say v_A, v_B, v_C, v_D) find two ears that includes all vertices in $\{v_0, v_1, v_2, \ldots, v_k\}$ for some $k \geq \min(q, 5)$ and $\{v_A, v_B, v_C, v_D\}$ such that one ear's end vertices are v_0 and v_1.*

The ear may include plundered segments from the heads of unprocessed paths. But, we do not plunder nodes from a DFS path that has exactly four vertices.

Lemma 2 (Induction Step for General Graphs). *Let D be a DFS path whose vertices are labeled $v_0, v_1, v_2, v_3, ..., v_q$ starting from the leaf vertex. Nodes v_0 and v_1 together represent the current 2-connected component, with v_1 as the hinging node for the remainder of the path. It is possible to do one of the following:*

1. *Find an ear with two end vertices v_0 and v_1 that includes all vertices in $\{v_0, v_1, v_2, \ldots, v_k\}$ for some $k \geq \min(q, 5)$.*
2. *Find two ears that includes all vertices in $\{v_0, v_1, v_2, \ldots, v_k\}$ for some $k \geq \min(q, 8)$ such that one ear's end vertices are v_0 and v_1.*

The ear may include plundered segments from the heads of unprocessed paths.

4.6 Finishing Step: Connecting to the Core

In the 2-ECSS problem, the core can be treated as an extra vertex, but one that does not have any vertex charge. But, in 2-VCSS, we have to ensure that the ear decomposition that we find hinges to the core at two different nodes. Therefore, we have to handle the last step with more care.

When 2-connecting a path to the core, if the algorithm reaches a stage when the last ear reaches the core, but leaves a cut vertex in it, we can add an extra

edge to cross it, and the extra edge can be paid with the charge we saved in the basis step. If there are up to 2 extra nodes between the current component and the core, we can show that the final ear can be completed without creating a cut vertex. Again, the first ear (basis step) generates a surplus charge of 2 and this pays for the extra edges needed. If there are three extra nodes between the core and the component, we show that either the final ear is completed without generating a cut vertex in the core or we find a witness. Since a witness set has an extra charge of $(2\alpha-2)$, three nodes have extra charge of $3(\alpha-1)$, we can pay for the additional edge needed to 2-connect it to core. We mention the finishing step lemma here. Proof is provided in the full paper.

Lemma 3 (Finishing Step). *Let O be the core and O' be the component. v_1, v_2, v_3 are the vertices between core and component. We can 2-connect O, O' and the three vertices in between using five edges or we can find a witness.*

In the finishing step we assume that core does not have any charge. This is true in all cases exept the case when core is just a node because the node has vertex charge. The case where core is a singleton node happens only once in the execution of the algorithm. This vertex charge is never used in the course of the algorithm and we save $4(\alpha - 1)$ from that vertex charge to find to find $\alpha Opt - 4(\alpha - 1)$ approximation algorithm.

5 Summary

In summary, we preprocess the graph to remove beta nodes, beta pairs, adjacent degree-2 nodes and nontrivial 2-cuts and apply our algorithm. Our algorithm uses a 3-cycle-free maximal 2-matching to select a suitable depth-first search tree of the graph. The DFS tree is broken into a collection of paths and we 2-connect the paths one at a time. We account for edges by a charging method where we try to pay using vertex charges whenever possible and show that when we need extra, path charges or witness charges are available to cover the deficit. Our algorithm finds a solution that uses at most $\frac{9}{7}(n+max\{|P|,|w|\})$ in general graphs and $\frac{5}{4}(n+max\{|P|,|w|\})$ in graphs with a minimum-degree of three at every node where $|P|$ is the number of paths and $|w|$ is the number of *witness* es in the given graph.

Theorem 1. *Given a 2-connected, undirected graph $G = (V, E)$, there is a polynomial-time algorithm that returns a 2-VCSS of G that is within $9/7 \approx 1.286$ of an optimal 2-VCSS.*

Theorem 2. *Given a 2-connected, undirected graph $G = (V, E)$ in which each node has degree at least 3, there is a polynomial-time algorithm that returns a 2-VCSS of G that is within $5/4 = 1.25$ of an optimal 2-VCSS.*

References

1. J. Cheriyan, A. Sebő, Z. Szigeti, *Improving on the 1.5 approximation of a smallest 2-edge connected spanning subgraph*, SIAM J. Discret. Math., **14**, pp. 170-180, 2001.
2. J. Cheriyan, S. Vempala and A. Vetta, *Approximation algorithms for minimum-cost k-connected subgraphs*, Proc. of the 34th ACM Symposium on the Theory of Computing (STOC), 2002.
3. J. Cheriyan and R. Thurimella, *Approximating minimum-size k-connected spanning subgraphs via Matching*, SIAM J. Comput., **30**, pp. 528-560, 2000.
4. C. G. Fernandes, *A better approximation for the minimum k-edge-connected spanning subgraph problem*, J. Algorithms, **28**, pp. 105-124, 1998.
5. H. N. Gabow, *An ear decomposition approach to approximating the smallest 3-edge connected spanning subgraph of a multigraph*, SIAM J. Disc. Math 18,1, 2004, pp.41-70.
6. H. N. Gabow, *Better performance bounds for finding the smallest k-edge connected spanning subgraph of a multigraph*, Proc. 14th Annual ACM-SIAM Symp. on Discret. Algorithms (SODA), 2003.
7. N. Garg, V. Santosh and A. Singla, *Improved approximation algorithms for biconnected subgraphs via better lower bounding techniques*, Proc. 4th Annual ACM-SIAM Symp. on Discret. Algorithms (SODA), pp. 103-111, 1993.
8. D. Hartvigsen, *Extensions of matching theory*, Ph.D. Thesis, Carnegie-Mellon University, 1984.
9. D. Hartvigsen, *The square-free 2-factor problem in bipartite graphs*, Proc. of the 7th Integer Programming and Combinatorial Optimization Conference (IPCO), pp. 234-241, 1999.
10. R. Jothi, B. Raghavachari, and S. Varadarajan, *A 5/4-approximation algorithm for minimum 2-edge-connectivity*, in Proc. 14th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 725-734, 2003.
11. S. Khuller and B. Raghavachari, *Improved approximation algorithms for uniform connectivity problems*, J. Algorithms, **21**, pp. 433-450, 1996.
12. S. Khuller and U. Vishkin, *Biconnectivity approximations and graph carvings*, J. Assoc. Comput. Mach., **41**, pp. 214-235, 1994.
13. P. Krysta and V. S. Anil Kumar, *Approximation algorithms for minimum size 2-connectivity problems*, Proc. 18th Intl. Symposium on Theoretical Aspects of Computer Science (STACS), pp. 431-442, 2001.
14. H. Nagamochi and T. Ibaraki. *Linear time algorithms for finding sparse k-connected spanning subgraph of a k-connected graph*, Algorithmica, **7**, pp. 583-596, 1992.
15. S. Vempala and A. Vetta, *Factor 4/3 approximations for minimum 2-connected subgraphs*, APPROX 2000, pp. 262-273, 2000.

Using Grammars to Generate Very Large Scale Neighborhoods for the Traveling Salesman Problem and Other Sequencing Problems*

Agustin Bompadre[1] and James B. Orlin[2]

[1] Operations Research Center, MIT, Cambridge, MA, USA
[2] Sloan School of Management, MIT, Cambridge, MA, USA
{abompadr, jorlin}@mit.edu

Abstract. Local search heuristics are among the most popular approaches to solve hard optimization problems. Among them, Very Large Scale Neighborhood Search techniques present a good balance between the quality of local optima and the time to search a neighborhood. We develop a language to generate exponentially large neighborhoods for sequencing problems using grammars. We develop efficient generic dynamic programming solvers that determine the optimal neighbor in a neighborhood generated by a grammar for sequencing problems such as the Traveling Salesman Problem or the Linear Ordering Problem. This framework unifies a variety of previous results on exponentially large neighborhood for the Traveling Salesman Problem and generalizes them to other sequencing problems.

1 Introduction

Very large scale neighborhood search has been used in a variety of contexts and employs a variety of techniques. For survey papers, see Ahuja et al [1] and Deĭneko and Woeginger [12]. Deĭneko and Woeginger [12] described a variety of techniques used to search exponential neighborhoods for the TSP in polynomial time. Other closely related papers include Burkard et al [7] and Ergun and Orlin [13]. Many of the efficient search techniques rely on dynamic programming recursions. In this paper, we unify these disparate results into a unifying framework based on context free grammars. We also provide a generic dynamic programming algorithm for finding the best tour in a grammar-induced neighborhood. When specialized to the neighborhoods given in Deĭneko and Woeginger [12], our generic algorithm achieves the same running time as the special purpose dynamic programs, except for the twisted sequence neighborhood. In that case, our generic DP improves upon the previous best bound by a factor of n. The framework developed for generating neighborhoods for TSP and the dynamic programming solver applies to other sequencing problems including the linear ordering problem.

* This research was supported in part by NSF grant DMI-0217123.

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 437–451, 2005.

For successful development of VLSN search, it will become increasingly important to find effective ways of describing the exponentially large neighborhood efficiently via a computer language. Here we have made substantial progress towards that goal in the case of sequencing problems.

1. We develop the first language for compactly generating exponentially large neighborhoods for sequencing problems. In fact, it is the first language for compactly generating exponentially large neighborhoods.
2. We develop the mathematical foundation for using regular grammars and context free grammars to describe very large neighborhoods for sequencing problems.
3. We develop a dynamic programming solver for the TSP that determines an optimum neighbor in time polynomial in the size of the problem and the number of rules. The solver uses the rules of the grammar as input as well as the TSP instance and current solution.
4. We develop dynamic programming solvers for other sequencing problems, such as the Linear Ordering Problem or the Minimum Latency Problem, that determine an optimum neighbor in time polynomial in the size of the problem and the number of rules. The solvers use the rules of the grammar as part of its input.
5. We provide efficient algorithms for enumerating the size of neighborhoods and for deciding whether two neighborhoods are distinct in the case that the generating grammars are context free and unambiguous.

The rest of the paper is organized as follows. In Section 2, we establish the notation and terminology for TSP, local search algorithms and grammars we use throughout the paper. In Section 3, we present neighborhoods defined in the previous literature and their description using grammars. In Section 4, we state the generic dynamic programming solver for finding the best tour in a grammar induced neighborhood, and we present algorithms to solve other problems for grammar based neighborhoods. In Section 5, we extend these results to a list of sequencing problems. Finally, we present our conclusions in Section 6.

2 Notation and Terminology

Traveling Salesman Problem. The Traveling Salesman Problem is to find the minimum distance tour on n cities $1, \ldots, n$. The distance from city i to city j is c_{ij}. We represent a tour using a permutation $\pi \in S_n$, where S_n is the set of permutations of $\{1, \ldots, n\}$. The permutation $\pi = (\pi(1), \ldots, \pi(n))$ refers to the tour in which the first city visited is $\pi(1)$, the next city visited is $\pi(2)$, and so on. We will also refer to π as a *tour*. The cost of the tour π is denoted as $c(\pi) = \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)} + c_{\pi(n)\pi(1)}$.

Neighborhoods. We describe neighborhoods for traveling salesman problems with n cities. For tour π, and permutation σ, we associate another tour

$\pi \circ \sigma = \pi(\sigma)$, which is formed by composing permutations π and σ. For example, if $\pi = (3, 2, 4, 1, 5)$ and if $\sigma = (2, 1, 3, 5, 4)$ then $\pi \circ \sigma = (2, 3, 4, 5, 1)$.

For a given subset N of tours and for a tour π, we let $N(\pi) = \{\pi \circ \sigma : \sigma \in N\}$, which is the neighborhood of tours π.

In the case that we are treating neighborhoods for different sized TSPs, let N_n denote the neighborhood set for problems with n cities. In the case that the number of cities is obvious from context, we drop the index, and denote the neighborhood set as N. In general, we assume that the identity permutation is in N_n, that is $(1, 2, \ldots, n) \in N_n$.

Very Large Scale Neighborhood Search. A neighborhood search algorithm starts with a feasible solution of the optimization problem and successively improves it by replacing it by an improved neighbor until it obtains a locally optimal solution. For many neighborhoods, an improving neighbor is determined by exhaustively enumerating all of the neighbors. We refer to a neighborhood as very large scale if the neighborhood is too large to be searched exhaustively and is searched using some more efficient search procedure.

In this paper, we are primarily (but not entirely) focused on neighborhoods with the following two properties, also given in Deĭneko and Woeginger [12].

1. *Exponential Size Property.* The neighborhood set N_n is exponentially large in n. That is, there is **no** polynomial function $f(n)$ such that $|N_n| = \mathrm{O}(f(n))$.
2. *Polynomially Searchable Property.* The neighborhood N_n may be searched in polynomial time. That is, there is an Algorithm A and a polynomial $f()$ such that for every tour $\pi \in S_n$, Algorithm A can find the minimum distance neighbor in $N_n(\pi)$ with respect to any cost function in time $\mathrm{O}(f(n))$.

Any neighborhood with the first property is referred to as an *exponential neighborhood*. Any neighborhood with the second property is said to be *searchable in polynomial time*.

Grammar Terminology. Roughly speaking, a grammar is a set of rules for how to compose strings in a language. In this subsection, we define the concepts from the theory of languages that are relevant in our work. We refer to the book by Hopcroft and Ullman [15] for more information on languages and grammars. We let $G = (V_{NT}, V_T, P, S)$ denote a grammar. V_{NT} is the set of variables or non-terminals, V_T is the set of terminals. We assume that V_{NT}, V_T are disjoint, and we let $V = V_{NT} \cup V_T$. A *string* α is a finite sequence of terminals and non-terminals. We let V^* be the set of all strings on V. The non-terminal $S \in V_{NT}$ is the *start* symbol. P is the set of production rules. A production rule $p \in P$ has the form

1. $A \rightarrow a_1 \ldots a_k B_1$ for some $k \geq 1$,
2. $A \rightarrow B_1 a_1 \ldots a_k$ for some $k \geq 1$,
3. $A \rightarrow B_1 B_2 \ldots B_k$ for some $k \geq 1$,

4. $A \to a_1 \dots a_k$ for some $k \geq 1$,
5. $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$,

with $A, B_1, \dots, B_k \in V_{NT}$, $a_1, \dots, a_k \in V_T$ and $\alpha_1, \alpha_2, \beta \in V^*$.

A grammar with production rules of this form is a *context-sensitive grammar*. A grammar with production rules of the form 1,2,3,4 is a *context-free grammar*. A grammar with production rules of the form 1 and 4 (2 and 4) is a *right (left) regular grammar*. A grammar with production rules of the form 1, 2, 3, 4 with at most two nonterminals on the right hand side is in *normal form*. A grammar with production rules of the form 1, 2, 4 is in *extended normal form*. We will not use production rules of the form 5, since a context-sensitive grammar is "too powerful". In particular, there is a way of generating all tours of n cities using n^{12} production rules in a context-sensitive grammar. One cannot search this neighborhood in polynomial time unless $P = NP$.

The *language generated* by G, denoted $L(G)$, is the set of strings, consisting solely of terminals, that can be derived from S.

Example 1. Let $G = (V_{NT}, V_T, P, S)$ be the grammar where $V_{NT} = \{S, B\}$, $V_T = \{a, b\}$ and $P = \{S \to aB, S \to ab, B \to Sb\}$. Then the language generated by G is $L(G) = \{ab, aabb, aaabbb, \dots\}$.

A grammar is *ambiguous* if one can generate the same string in two distinct ways using production rules. It is *unambiguous* otherwise. Example 1 is unambiguous.

TSP Grammar. Let $G = (V_{NT}, V_T, P, S)$ be a regular or context-free grammar that generates a finite language. We refer to G as a *traveling salesman tour grammar on n cities* if every string of terminals generated by the grammar corresponds to a TSP tour. For regular and context free grammars, the generic TSP grammar has the following properties: The set of terminals contains n symbols (i.e., $V_T = \{a_1, \dots, a_n\}$), each nonterminal on the right-hand side of a production rule appears on the left-hand side of some production rule and there exists a function $Cities : V_{NT} \cup V_T \to 2^{\{1,\dots,n\}}$ that assigns a subset of cities to each element of V with the following properties:

1. $Cities(S) = \{1, \dots, n\}$,
2. $Cities(A) = \cup_{j=1}^{k} Cities(R_j)$ for each production rule $A \to R_1 R_2 \dots R_k$,
3. $Cities(R_i) \cap Cities(R_j) = \emptyset$ for each production rule $A \to R_1 R_2 \dots R_k$, for $1 \leq i < j \leq k$,
4. $Cities(a_j) = \{j\}$ for every terminal $a_j \in V_T$.

We extend the definition of *Cities* to strings generated by G. We observe that $Cities(\alpha)$ is uniquely defined for any string α. We observe that a unique subtour $(i_1, i_2, \dots, i_r)$ is associated to each string of terminals $\alpha = a_{i_1} \dots a_{i_r}$ generated by G. We refer to any grammar satisfying the above properties as a canonical TSP grammar.

Proposition 1. *A canonical TSP Grammar generates a non-empty language. Every element of the language is a tour on n cities.*

Typically we will omit the word canonical and just say TSP grammar. The neighborhood set N generated by the TSP grammar G is the language $L(G)$. The neighborhood set of tour π generated by G, is the set $N(\pi) = \{\pi\alpha : \alpha \in L(G)\}$.

We extend the definition of ambiguity to TSP grammars. We distinguish permutations that are different even if they represent the same tour.

Definition 1. *A TSP grammar (G, Cities) is ambiguous if one can generate the same permutation in two distinct ways using production rules. It is unambiguous otherwise.*

3 Neighborhoods Generated by TSP Neighborhoods Grammars

Any neighborhood with K neighbors is trivially generated using K production rules. We are primarily interested when the size of the neighborhood is exponential in the number of rules.

3.1 Exponential Neighborhoods

We next describe a number of exponential neighborhoods for the TSP. Most of them are covered in the survey paper by Deĭneko and Woeginger [12] and the papers by Burkard et al [7] and Ergun and Orlin [13]. Their common feature is that they can be optimized efficiently using a Dynamic Programming approach. In the following subsection we give polynomial length descriptions of these neighborhoods using grammars.

Pyramidal Neighborhood. The Pyramidal neighborhood consists of *pyramidal permutations*, where the city labels increase monotonically to n and then decrease. That is,

$$N_P = \left\{ \sigma : \begin{array}{l} \sigma(1) < \sigma(2) < \ldots < \sigma(k) = n, \\ \sigma(k+1) > \sigma(k+2) > \ldots > \sigma(n) \text{ for some } k \geq 1 \end{array} \right\}.$$

For example, $(1, 3, 4, 6, 5, 2)$ is a pyramidal permutation whereas $(1, 3, 6, 2, 4, 5)$ is not.

Klyaus [16] gave an $O(n^2)$ dynamic programming algorithm for finding the best pyramidal tour. See also Gilmore, Lawler, and Schmoys [14]. Sarvanov and Doroshko [18] employed the concept of pyramidal tours for use in very large scale neighborhood search. Carlier and Villon [9] combined the pyramidal tour neighborhood with a cyclic shift neighborhood in very large scale neighborhood search. They determined empirically that this composite neighborhood performed far better than 2-opt.

Permutation Tree Neighborhood. We give the description by Deĭneko and Woeginger [12]. A permutation tree T over $\{1, \ldots, n\}$ is a rooted, ordered tree that satisfies the following.

1. There is a bijective mapping between leaves of T and cities.
2. An interior node has at least two sons.
3. An interior node i with d sons has associated a set of permutations $\Phi_i \subseteq S_d$

A permutation tree T defines a set of permutations N_T as follows. For each node $i \in T$, let T_i be the subtree of T rooted at i. We associate the set N_{T_i} of sequences of cities to each subtree T_i. If i is a leaf of T then $N_{T_i} = \{i\}$. If i is an interior node of T with d sons $i_1, \ldots, i_d$ then $N_{T_i} = \cup_{\phi \in \Phi_i} N_{T_{i_{\phi(1)}}}, \ldots, N_{T_{i_{\phi(d)}}}$. The set of permutations N_T is the set N_{T_r} defined by the root node $r \in T$.

Booth and Lueker [6] analyzed a special case of permutation trees that they called PQ-trees. Burkard, Deĭneko, and Woeginger [8] investigated finding the best tour in the PQ- tree neighborhood, which was later generalized by Deĭneko and Woeginger [12] to include all permutation trees. They also pointed out that pyramidal tours are a special case of permutation trees.

Twisted Sequences. Aurenhammer [2] described permutations in terms of possible derivations from permutation trees. A permutation tree T is a *twisted tree* if it satisfies the following conditions.

1. The identity sequence $\mathrm{id}_n = (1, \ldots, n) \in N_T$.
2. Every interior node i with d sons has associated a set of permutations $\Phi_i = \{\mathrm{id}_d, \mathrm{id}_d^-\}$, where id_d^- is the identity permutation in reverse order.

A permutation σ is a *twisted sequence* if it is generated by a twisted tree. That is, $N_{\mathrm{Twisted}} = \{\sigma : \sigma \in N_T \text{ for some twisted tree } T\}$.

For example, $(3, 2, 4, 1)$ is a twisted sequence whereas $(1, 3, 5, 2, 4, 6)$ is not.

Dynasearch Neighborhood. Potts and van de Velde [17] defined this class of neighborhoods. A *Dynasearch Neighborhood* is obtained by combining a set of independent simple moves such as 2-exchanges, swaps, or insertions. Two moves on the TSP are said to be *independent* if the subpaths they are modifying contain no common edges. For example, consider two 2-exchange moves affecting subpaths $i, i+1, \ldots, j$ and $k, k+1, \ldots, l$ by breaking edges $(i, i+1), (j-1, j)$ and $(k, k+1), (l-1, l)$ and adding edges $(i, j-1), (i+1, j)$ and $(k, l-1), (k+1, l)$. These two moves are independent if and only if either $j < k$ or $l < i$. Based on 2-exchange moves, the sequence $(1, 5, 3, 4, 2, 6, 9, 8, 7)$ is a Dynasearch sequence, whereas $(1, 5, 9, 4, 2, 6, 7, 8, 3)$ is not. Based on 2-exchange the following

Balas and Simonetti Neighborhood. Balas and Simonetti [4] investigated a very large scale neighborhood for the TSP which we denote by N_{BS}^k. We note that in their paper they define a solution of the TSP as a permutation of places instead of permutation of cities. We give the definition of N_{BS}^k in terms of permutation of cities. Let k be a parameter. They consider all tours such that city 1 is visited first and that city i precedes city j whenever $i + k \leq j$ for some fixed parameter k. That is, $N_{BS}^k = \{\sigma : \sigma(1) = 1 \text{ and } \sigma^{-1}(i) < \sigma^{-1}(j) \text{ for } i + k \leq j\}$. When $k = 3$, the sequence $(1, 4, 2, 5, 3, 6)$ belongs to N_{BS}^3 whereas $(1, 6, 2, 3, 4, 5)$ does not.

3.2 Exponential Neighborhoods as TSP Neighborhood Grammars

In this subsection we give grammars with a polynomial number of rules that yield the exponential sized neighborhoods of Subsection 3.1. In each case, we state the correspondence without proof. In each case we will state the values for *Cities*(); however that is for the value of the reader and is not a formal part of the grammar.

Pyramidal Tours. The following traveling salesman tour grammar generates the Pyramidal Neighborhood.

$V_{NT} = \{S\} \cup \{A_{j,n} : 1 \leq j \leq n\}$, $V_T = \{1, \ldots, n\}$. The value of *Cities* on each terminal and non-terminal is $\mathit{Cities}(S) = \{1, \ldots, n\}$, $\mathit{Cities}(A_{j,n}) = \{j, \ldots, n\}$ for $1 \leq j \leq n$, $\mathit{Cities}(j) = \{j\}$ for $1 \leq j \leq n$. Its production rules are

The Pyramidal Neighborhood Grammar
$S \rightarrow A_{1,n}$,
$A_{j,n} \rightarrow j, A_{j+1,n}$, for $1 \leq j \leq n-1$
$A_{j,n} \rightarrow A_{j+1,n}, j$, for $1 \leq j \leq n-1$
$A_{n,n} \rightarrow n$.

Permutation Trees and Their Grammars. A TSP neighborhood grammar G is tree-based if for any non-terminals A, B then either

- $\mathit{Cities}(A) \cap \mathit{Cities}(B) = \emptyset$, or
- $\mathit{Cities}(A) \subset \mathit{Cities}(B)$, or
- $\mathit{Cities}(B) \subset \mathit{Cities}(A)$.

We can represent any tree-based grammar as a tree, where each non-terminal is an internal node of the tree; each terminal is a leaf of the tree. Also, there is an arc from R to R' if $\mathit{Cities}(R) \subset \mathit{Cities}(R')$ and there is no other non-terminal R'' with $\mathit{Cities}(R) \subset \mathit{Cities}(R'') \subset \mathit{Cities}(R')$. Such a tree is called a *permutation tree*, as per Deĭneko and Woeginger [12].

Twisted Sequence Neighborhood. We describe a simply stated grammar that generates all twisted sequences. Let the set of non-terminals and the set of terminals be $V_{NT} = \{S\} \cup \{A_{i,j} : 1 \leq i \leq j \leq n\}$, $V_T = \{1, \ldots, n\}$ respectively. The value of *Cities* on each terminal and non-terminal is $\mathit{Cities}(S) = \{1, \ldots, n\}$, $\mathit{Cities}(A_{i,j}) = \{i, \ldots, j\}$ for $1 \leq i \leq j \leq n$, $\mathit{Cities}(j) = \{j\}$ for $1 \leq j \leq n$. Its production rules are

Twisted Sequence Neighborhood Grammar
$S \rightarrow A_{1,n}$,
$A_{i,j} \rightarrow i, \ldots, j$ for $1 \leq i \leq j \leq n$,
$A_{i,j} \rightarrow A_{i,k}, A_{k+1,j}$ for $1 \leq i \leq k < j \leq n$,
$A_{i,j} \rightarrow A_{k+1,j}, A_{i,k}$ for $1 \leq i \leq k < j \leq n$.

Restricted Dynamic Programs. We next describe a TSP neighborhood grammar for generating all the permutations whose initial city is city 1, which we refer

to as the "complete TSP neighborhood grammar". This particular grammar has $n2^{n-1}$ different production rules, and generates $(n-1)!$ permutations. We then give polynomially bounded subsets of the grammar in which the neighborhood is exponential.

Complete TSP Neighborhood Grammar
$S \to A_{\{1,\ldots,n\}}$,
$A_R \to A_{R\backslash j}, j$ for each $j \in R\backslash 1$, and for each $R \subseteq \{1,\ldots,n\}$ with $1 \in R$,
$A_{\{1\}} \to 1$.

The value of *Cities* on each terminal and non-terminal is as follows. $Cities(S) = \{1,\ldots,n\}$, $Cities(A_R) = R$ for each subset $R \subseteq \{1,\ldots,n\}$ with $1 \in R$, $Cities(j) = \{j\}$ for $1 \le j \le n$.

In the following, let $\mathcal{S}$ be any collection of subsets of $\{1,\ldots,n\}$ with the property that for each $R \in \mathcal{S}$, $1 \in R$. We now create a subset of the Complete TSP grammar, by restricting production rules to involve only non-terminals corresponding to sets in $\mathcal{S}$.

Restricted TSP Neighborhood Grammar
$S \to A_{\{1,\ldots,n\}}$,
$A_R \to A_{R\backslash j}, j$ for each $R \in \mathcal{S}$, and for each $j \in R$ such that $R\backslash j \in \mathcal{S}$,
$A_{\{1\}} \to 1$.

Balas and Simonetti Neighborhood. Let $\mathcal{S}_{BS} := \{R \subseteq \{1,\ldots,n\} : \text{ if } i \in R \text{ and } j \notin R, \text{ then } i-k < j\}$. The Balas-Simonetti neighborhood corresponds to the restricted TSP Neighborhood grammar with $\mathcal{S}$ replaced by $\mathcal{S}_{BS}$.

Dynasearch Neighborhoods. The following Dynasearch neighborhood is based on 2-exchange. The value of *Cities* on each terminal and non-terminal is as follows. $Cities(S) = \{1,\ldots,n\}$, $Cities(A_{i,j}) = \{i,\ldots,j\}$ for $1 \le i \le j-1 \le n-1$, $Cities(j) = \{j\}$ for $1 \le j \le n$.

The Dynasearch 2-exchange Neighborhood Grammar
$S \to A_{1,n}$,
$A_{1,j} \to A_{1,i-1}, A_{i,j}$ for $2 \le i \le j \le n$,
$A_{i,j} \to i,\ldots,j$ for $1 \le i \le j-1 \le n-1$,
$A_{i,j} \to i, j-1, j-2,\ldots,i+2, i+1, j$ for $1 \le i \le j-3 \le n-1$.

3.3 Compounded Neighborhoods

Two grammars $G = (V_{NT}, V_T, P, S)$ and $G' = (V'_{NT}, V'_T, P', S)$ can be combined to generate a *compounded* grammar $G \cup G' := (V_{NT} \cup V'_{NT}, V_T \cup V'_T, P \cup P', S)$. The number of production rules of the compounded grammar is at most the sum of the number of production rules of G, G'. The neighborhood generated by the compounded grammar includes the neighborhood generated by G and G', and possibly much more. It can be searched efficiently, as Theorem 1 shows. As

an example, we describe the neighborhood obtained by compounding Pyramidal and Dynasearch grammars.

Pyramidal and Dynasearch Compounded Neighborhoods. The following neighborhood is obtained by compounding the Pyramidal and the Dynasearch 2-exchange grammars. The value of *Cities* on each terminal and non-terminal is $Cities(S) = \{1, \ldots, n\}$, $Cities(A_{i,j}) = \{i, \ldots, j\}$ for $1 \leq i \leq j - 1 \leq n - 1$, $Cities(j) = \{j\}$ for $1 \leq j \leq n$.

The Pyramidal and Dynasearch 2-exchange Neighborhood Compounded Grammar

$S \rightarrow A_{1,n}$,
$A_{j,n} \rightarrow j, A_{j+1,n}$, for $1 \leq j \leq n-1$
$A_{j,n} \rightarrow A_{j+1,n}, j$, for $1 \leq j \leq n-1$
$A_{1,j} \rightarrow A_{1,i-1}, A_{i,j}$ for $2 \leq i \leq j \leq n$,
$A_{i,j} \rightarrow i, \ldots, j$ for $1 \leq i \leq j - 1 \leq n - 1$,
$A_{i,j} \rightarrow i, j-1, j-2, \ldots, i+2, i+1, j$ for $1 \leq i \leq j - 3 \leq n - 1$.

4 Algorithms

4.1 A Generic Solution Procedure Using Dynamic Programming

We present a dynamic programming algorithm for finding the best tour in a neighborhood generated by a grammar G. The initial tour is $\pi = (\pi(1), \ldots, \pi(n))$. We denote by $f(i, j)$ the arc cost $c_{\pi(i),\pi(j)}$. We introduce some additional notation. Given a terminal or non terminal R and cities $i, j \in Cities(R)$, we denote by $V(i, R, j)$ the minimum cost path from $\pi(i)$ to $\pi(j)$ passing through all the cities of $\pi(Cities(R))$ as generated by the grammar. Given a terminal or non terminal R and cities $i \in Cities(R), j \notin Cities(R)$ we denote by $\bar{V}(i, R, j)$ the minimum cost path from $\pi(i)$, passing through all the cities of $\pi(Cities(R))$ as generated by the grammar, and ending at city $\pi(j)$. Similarly, given a production rule $p : A \rightarrow R_1 R_2 \ldots R_r$ and cities $i, j \in Cities(A)$, we denote by $V(i, p, j)$ the minimum cost path from $\pi(i)$ to $\pi(j)$ passing through all the cities of $\pi(Cities(A))$ as generated by p.

The best tour generated by the grammar has value $\min_{i,j}\{V(i, S, j) + f(j, i)\}$. The following is a dynamic programming recursion for computing the best tour in a TSP Neighborhood grammar.

1. If a is a terminal with $Cities(a) = \{k\}$, then $V(k, a, k) = 0$.
2. Suppose p is the production rule $A \rightarrow a_1 a_2 \ldots a_r B$. Let $Cities(a_k) = \{i_k\}$ for $1 \leq k \leq r$. Then, for $j \in Cities(B)$:
   $V(i_1, p, j) = \min\{\sum_{k=1}^{r-1} f(i_k, i_{k+1}) + f(i_r, t) + V(t, B, j) : t \in Cities(B)\}$.
3. Suppose p is the production rule $A \rightarrow R_1, R_2, \ldots, R_r$.
   Then, for $k = 2$ to r and for $i \in Cities(R_1), j \in Cities(R_k)$:
   $V(i, R_1, R_2, \ldots, R_k, j) =$

$\min\{\bar{V}(i, R_1, R_2, \ldots, R_{k-1}, t) + V(t, R_k, j) : t \in \mathit{Cities}(R_k)\}$,
For $k = 1$ to $r - 1$ and for $i \in \mathit{Cities}(R_1), j \in \mathit{Cities}(R_{k+1})$:
$\bar{V}(i, R_1, R_2, \ldots, R_k, j) =$
$\min\{V(i, R_1, R_2, \ldots, R_k, s) + f(s, j) : s \in \mathit{Cities}(R_k)\}$,
For $i, j \in \mathit{Cities}(A)$:
$V(i, p, j) = V(i, R_1, R_2, \ldots, R_r, j)$

4. If R is a non-terminal, then
For $i, j \in \mathit{Cities}(R)$:
$V(i, R, j) = \min\{V(i, p, j) : p$ is a production rule and p applies to $R\}$.

Proposition 2. *Suppose that p is the production rule $p : A \to R_1 R_2 \ldots R_r$. Then the time to compute $V(i, p, j)$ for all $i, j \in \mathit{Cities}(A)$ from the values for $R_1, R_2, \ldots, R_r$ is $O(n^3)$.*

The proof of the following theorem follows from this proposition.

Theorem 1. *Let K be the number of production rules of a TSP neighborhood grammar for a problem with n cities. Then the time to compute the best neighbor generated by the grammar is $O(Kn^3)$. If the grammar is in extended normal form, the time is $O(Kn^2)$.*

With further analysis in special cases, the running time can sometimes be improved. When specialized to the neighborhoods defined in Section 3, the running time of the generic DP algorithm matches the running times from the literature. It also improves by a factor of n the running time to solve the Twisted Neighborhood given in [12]. Table 1 shows the running times on these neighborhoods.

Table 1. Running times on TSP grammars for the generic dynamic programming algorithm

TSP Neighborhood Grammar	**Running time**
2-opt Neighborhood	O(^{2})
Pyramidal Tour Neighborhood	O(^{2})
Tree-based grammars with at most F production rules per non-terminal	O(F ^{3})
Twisted Sequence Neighborhood	O(^{6})
Complete Neighborhood	O($^2 2^n$)
Balas-Simonetti Neighborhood	O($k2^k$)
Dynasearch Neighborhoods	O(^{2})

4.2 Counting the Neighbors and Complexity Questions

We address the following questions for unambiguous TSP grammars:

1. Can we count the number of tours in the neighborhood?
2. Given two TSP grammars, do they generate different neighborhoods?
3. Is a TSP grammar symmetric (i.e., $\pi' \in N(\pi)$ iff $\pi \in N(\pi')$)?

In order to answer these questions, we associate a polynomial to a TSP grammar G, defined as follows. For each $1 \leq i, j \leq n$, a variable $Y_{i,j}$ is associated to the placement of city j in the *i-th* position. Let Y denote the vector of variables $Y_{i,j}$. Each permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$ has associated the monomial $g_\sigma(Y) = \prod_{i=1}^{n} Y_{i,\sigma(i)}$, and each TSP grammar G has associated the polynomial $g_G(Y) = \sum_{\sigma \in L(G)} g_\sigma(Y)$. When the TSP grammar is unambiguous, this polynomial can be evaluated efficiently on any vector Y by means of recursive formulas. The polynomial $g_G(Y)$ satisfies a number of properties. The size of the neighborhood generated by G is equal to $g_G(1)$. The associated polynomials of different grammars are different, and they give different results with high probability when evaluated in a random point (see Schwartz [19]). The following two theorems hold.

Theorem 2. *Given an unambiguous TSP grammar G with n cities and K production rules, we can compute $|L(G)|$ in $O(Kn^2)$ time. Given two unambiguous TSP grammars G_1, G_2 with n cities and K_1, K_2 production rules respectively, there exists a Monte Carlo algorithm that checks if $L(G_1) \neq L(G_2)$ in $O((K_1 + K_2)n^2)$ time. That is, if $L(G_1) = L(G_2)$ the algorithm answers correctly. If $L(G_1) \neq L(G_2)$ the algorithm answers correctly with probability greater than $1/2$.*

Theorem 3. *Given an unambiguous TSP grammar G with n cities and K production rules, there exists a Monte Carlo algorithm that checks if $L(G)$ is symmetric. That is, if $L(G)$ is symmetric the algorithm answers correctly. If $L(G)$ is not symmetric the algorithm answers correctly with probability greater than $1/2$. It runs in $O(Kn^2)$ time.*

5 Extensions to Sequencing Problems

The neighborhoods defined by grammars can be used in other sequencing problems. In this section we describe the Linear Ordering Problem, the Minimum Latency Problem, and a Weighted Bipartite Matching Problem with Side Constraints, for which the neighborhoods generated by grammars can be searched efficiently.

Linear Ordering Problem. The cost of a permutation $\pi \in S_n$ is $\sum_{i<j} c_{\pi(i)\pi(j)}$. The Linear Ordering Problem is to find the permutation $\pi \in S_n$ with minimum cost. This problem has applications in voting and economics.

We next describe the dynamic programming algorithm to find the optimal solution in a grammar-based neighborhood. The initial permutation is $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$. We denote by $f(i,j)$ the cost $c_{\pi(i),\pi(j)}$. Given a terminal or non terminal R, we denote by $V(R)$ the value $\min\{\sum_{i,j \in R: i<j} f(\sigma(i), \sigma(j)) : \sigma \in L(G)\}$ that corresponds to the best ordering of elements in R as generated by the grammar. Similarly, given a production rule $p : A \rightarrow R_1 R_2 \ldots R_r$ we denote by $V(p)$ the cost of the best ordering of elements in R as generated by the grammar. The following is a dynamic programming recursion for computing the best permutation in a Neighborhood grammar.

1. If a is a terminal, then $V(a) = 0$.
2. Suppose p is the production rule $A \rightarrow R_1, R_2, \ldots, R_r$.
   Then, $V(p) = \sum_{1 \leq t_1 < t_2 \leq r} \sum_{i \in R_{t_1}, j \in R_{t_2}} f(i,j) + \sum_{k=1}^{r} V(R_k)$
3. If R is a non-terminal, then
   $V(R) = \min\{V(p) : p$ is a production rule and p applies to $R\}$.

The following theorem states the complexity of searching a neighborhood for the Linear Ordering Problem.

Theorem 4. *Let K be the number of production rules of a neighborhood grammar for a linear ordering problem with n elements. Then the time to compute the best neighbor generated by the grammar is $O(Kn^2)$.*

Minimum Latency Problem. The Minimum Latency Problem is to find a tour through n customers (cities) with minimum latency time. To be more precise, a permutation $\pi \in S_n$ represents a tour through n cities. For a given tour π, each city $\pi(i)$ experience a latency $c(\pi(i)) := \sum_{k=1}^{i-1} c_{\pi(k),\pi(k+1)}$. The cost of a permutation $\pi \in S_n$ of cities is the sum of costs that each city experiences: $c(\pi) := \sum_{k=1}^{n-1} c(\pi(k)) = \sum_{k=1}^{n-1} (n-k) c_{\pi(k)\pi(k+1)}$. The Minimum Latency Problem is to find a permutation $\pi \in S_n$ with minimum cost (see Blum et al [5]).

The initial tour is $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$. We denote by $f(i,j)$ the arc cost $c_{\pi(i),\pi(j)}$. Given a subsequence of cities $a_b, \ldots, a_e$ located in positions $b, b+1, \ldots, e$, we denote by $V(b, a_b, \ldots, a_e) = \sum_{k=b}^{e} (n-k+1) c_{a_k, a_{k+1}}$ the latency cost associated to it. Given a terminal or non terminal R, cities $i, j \in \mathit{Cities}(R)$, and an initial position $1 \leq b \leq n$, we denote by $V(b, i, R, j)$ the minimum latency cost of a path that starts from $\pi(i)$ in position b, ends in city $\pi(j)$ and visits all the cities of $\pi(\mathit{Cities}(R))$, as generated by the grammar. Given a terminal or non terminal R, cities $i \in \mathit{Cities}(R), j \notin \mathit{Cities}(R)$, and an initial position $1 \leq b \leq n$, we denote by $\bar{V}(b, i, R, j)$ the minimum latency cost of a path that starts from $\pi(i)$ in position b, visits all the cities of $\pi(\mathit{Cities}(R))$ as generated by the grammar, and ends in city $\pi(j)$. Similarly, given a production rule $p : A \rightarrow R_1 R_2 \ldots R_r$, cities $i, j \in \mathit{Cities}(A)$, and an initial position $1 \leq b \leq n$, we denote by $V(b, i, p, j)$ the minimum cost path that starts from city $\pi(i)$ in position b, ends in city $\pi(j)$ and visits all the cities of $\pi(\mathit{Cities}(A))$ as generated by p.

The best tour generated by the grammar has value $\min_{i,j}\{V(1, i, S, j)\}$. The following is a dynamic programming recursion for computing the best tour in a Minimum Latency Problem Neighborhood grammar.

1. If a is a terminal with $\mathit{Cities}(a) = k$, then for $1 \leq b \leq n$,
   $V(b, k, a, k) = 0$.
2. Suppose p is the production rule $A \rightarrow a_1 a_2 \ldots a_r B$. Let $\mathit{Cities}(a_k) = \{i_k\}$ for $1 \leq k \leq r$. Then, for $j \in \mathit{Cities}(A), 1 \leq b \leq n$:
   $V(b, i_1, p, j) = \min\{\sum_{k=1}^{r-1} (n+1-b-k) f(i_k, i_{k+1}) + (n+1-b-r) f(i_r, t) + V(b+r, t, B, j) : t \in \mathit{Cities}(B)\}$.
3. Suppose p is the production rule $A \rightarrow R_1, R_2, \ldots, R_r$.
   Then, for $k = 2$ to r and for $i \in \mathit{Cities}(R_1), j \in \mathit{Cities}(R_k), 1 \leq b \leq n$:

$V(b, i, R_1, R_2, \ldots, R_k, j) =$
$\min\{\bar{V}(b, i, R_1, R_2, \ldots, R_{k-1}, t) + V(b + \sum_{l=1}^{k-1} |R_l|, t, R_k, j) : t \in \mathit{Cities}(R_k)\}$,
For $k = 1$ to $r - 1$ and for $i \in \mathit{Cities}(R_1), j \in \mathit{Cities}(R_{k+1}), 1 \leq b \leq n$:
$\bar{V}(b, i, R_1, R_2, \ldots, R_k, j) =$
$\min\{V(b, i, R_1, R_2, \ldots, R_k, s) + (n - b - \sum_{l=1}^{k} |R_l| + 1) f(s, j) : s \in \mathit{Cities}(R_k)\}$,
For $i, j \in \mathit{Cities}(A), 1 \leq b \leq n$:
$V(b, i, p, j) = V(b, i, R_1, R_2, \ldots, R_r, j)$

4. If R is a non-terminal, then
For $i, j \in \mathit{Cities}(R), 1 \leq b \leq n$:
$V(b, i, R, j) = \min\{V(b, i, p, j) : p$ is a production rule and p applies to $R\}$.

The following theorem holds.

Theorem 5. *Let K be the number of production rules of a grammar for the minimum latency problem with n cities. The time to compute the best neighbor generated by the grammar is $O(Kn^4)$. The running time on Left Regular Grammars is $O(Kn^2)$.*

Weighted Bipartite Matching Problem with Side Constraints. We describe the following timetabling problem. Let $L = \{1, \ldots, n\}$ be a set of lessons, and for each $l \in L$, let b_l be the number of consecutive time slots lesson l needs. Let $T = \{t_1, \ldots, t_m\}$ be the set of available time slots. A time slot can be used by at most one lesson. For each lesson l and each time slot t, let c_{lt} be the cost of assigning time slots $t, t + 1, \ldots, t + b_l - 1$ to lesson l. The Weighted Bipartite Matching Problem with side constraints (WBMPSC) is to find a feasible matching with minimum cost. If $b_l = 1$ for all lesson l then the problem is a (regular) weighted matching and thus is solvable in polynomial time. However, even if $b_l \leq 2$ the problem becomes $NP-$hard (see ten Eikelder and Willemen [20]).

If one knows the sequence of lessons, one can directly compute the objective value. So, the WBMPSC is another sequencing problem.

Theorem 6. *Let K be the number of production rules of a grammar for the WBMPSC, with n lessons and m time slots. Then the time to compute the best neighbor generated by the grammar is $O(Knm^3)$.*

6 Conclusions

We present a language to generate exponentially large neighborhoods for sequencing problems using grammars. This is the first language proposed for generating exponentially large neighborhoods. We develop generic dynamic programming solvers that determine the optimal neighbor in a neighborhood generated by a grammar for the Traveling Salesman Problem, the Linear Ordering Problem, the Minimum Latency Problem, or a Weighted Bipartite Matching Problem with Side Constraints, in time polynomial in the size of the problem

and the number of rules of the grammar. Using this framework we unify a variety of previous results in defining and searching efficiently exponentially large neighborhoods for the traveling salesman problem. We also present efficient algorithms for enumerating the size of neighborhoods and for deciding whether two neighborhoods are distinct in the case that the generating grammars are context free and unambiguous.

Acknowledgments. We would like to thank Moshe Dror for fruitful discussions on compounded neighborhoods. We also wish to acknowledge partial support through NSF contract DMI-0217123.

References

1. R. K. Ahuja, O. Ergun, J. B. Orlin, A. P. Punnen (2002). "A survey of very large-scale neighborhood search techniques". Discrete Applied Mathematics 123, 75-102.
2. F. Aurenhammer (1998). "On-line sorting of twisted sequences in linear time". BIT 28, 194-204.
3. E. Balas (1996). "New classes of efficiently solvable generalized traveling salesman problem". Annals of Operations Research 86, 529-558.
4. E. Balas, N. Simonetti (2001). "Linear Time Dynamic Programming Algorithms for New Classes of Restricted TSPs: A Computational Study". INFORMS Journal on Computing 13:1, 56-75.
5. A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, M. Sudan (1994). "The minimum latency problem". STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, 163-171.
6. K. S. Booth, G. S. Lueker (1976). "Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms". J. Comput. Syst. Sci. 13, 335-379.
7. R. E. Burkard, V. G. Deĭneko, R. van Dal, J. A. A. van der Veen, G. J. Woeginger (1998). "Well-Solvable Special Cases of the Traveling Salesman Problem: A Survey". SIAM Review 40, 3 496-546.
8. R. E. Burkard, V. G. Deĭneko, G. J. Woeginger (1998). "The traveling salesman probem and the PQ-tree". Mathematics of Operations Research 23, 613-623.
9. J. Carlier, P. Villon (1990). "A new heuristic for the traveling salesman problem". RAIRO - Oper. Res. 24, 245-253.
10. R. K. Congram (2000). "Polynomially searchable exponential neighborhoods for sequencing problems in combinatorial optimisation". Ph.D. thesis, University of Southampton, UK.
11. R. K. Congram, C. N. Potts, S. L. van de Velde (2002). "An Iterated Dynasearch Algorithm for the Single-Machine Total Weighted Tardiness Scheduling Problem". INFORMS Journal on Computing 14, 52-67.
12. V. G. Deĭneko, G. J. Woeginger (2000). "A study of exponential neighborhoods for the Traveling Salesman Problem and for the Quadratic Assignment Problem". Mathematical Programming, Ser. A, 78, 519-542.
13. Ö. Ergun and J. B. Orlin (2003). "Dynamic Programming Methodologies in Very Large Scale Neighborhood Search Applied to the Traveling Salesman Problem". MIT Sloan Working Paper No. 4463-03.

14. P. C. Gilmore, E. L. Lawler, D. B. Schmoys (1985). "Well solved special cases". Chapter 4 in *The traveling salesman problem.* John Wiley, Chichester.
15. J. E. Hopcroft, J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Publishing Company, 1979.
16. P. S. Klyaus (1976). "The structure of the optimal solution of certain classes of traveling salesman problems". Vestsi Akad. Nauk BSSR, Phys. Math. Sci., Minsk, 95-98 (in Russian).
17. C. N. Potts, S. L. van de Velde (1995). "Dynasearch–Iterative local improvement by dynamic programming. Part I. The traveling salesman problem", Technical Report, University of Twente, The Netherlands.
18. V. I. Sarvanov, N. N. Doroshko (1981). "The approximate solution of the traveling salesman problem by a local algorithm that searches neighborhoods of exponential cardinality in quadratic time". Software: Algorithms and Programs 31, Mathematical Institute of the Belorrusian Academy of Sciences, Minsk, 8-11 (in Russian).
19. J. T. Schwartz (1980). "Fast Probabilistic Algorithms for Verification of Polynomial Identities", Journal of the ACM (JACM) 27, Issue 4, 701 - 717.
20. H.M.M. ten Eikelder, R.J. Willemen (2001). "Some Complexity Aspects of Secondary School Timetabling Problems", Proceedings of PATAT 2000, 87:18-27.

A Study of Domino-Parity and k-Parity Constraints for the TSP

William Cook, Daniel Espinoza, and Marcos Goycoolea

ISYE, Georgia Institute of Technology

Abstract. Letchford (2000) introduced the domino-parity inequalities for the symmetric traveling salesman problem and showed that if the support graph of an LP solution is planar, then the separation problem can be solved in polynomial time. We generalize domino-parity inequalities to multi-handled configurations, introducing a superclass of bipartition and star inequalities. Also, we generalize Letchford's algorithm, proving that for a fixed integer k, one can separate a superclass of k-handled clique-tree inequalities satisfying certain connectivity characteristics with respect to the planar support graph. We describe an implementation of Letchford's algorithm including pruning methods to restrict the search for dominoes, a parallelization of the main domino-building step, heuristics to obtain planar-support graphs, a safe-shrinking routine, a random-walk heuristic to extract additional violated constraints, and a tightening procedure to allow us to modify existing inequalities as the LP solution changes. We report computational results showing the strength of the new routines, including the optimal solution of the TSPLIB instance pla33810.

1 Introduction

Let $G = (V, E)$ be a complete graph with edge costs $(c_e : e \in E)$. The symmetric traveling salesman problem, or TSP, is to find a minimum-cost tour in G, that is, a Hamiltonian cycle of minimum total edge cost. A tour can be represented as a 0-1 vector $x = (x_e : e \in E)$, where $x_e = 1$ if edge e is used in the tour and $x_e = 0$ otherwise. In the Dantzig, Fulkerson, and Johnson [7] cutting-plane method for the TSP, a linear programming (LP) relaxation is created by iteratively finding linear inequalities that are satisfied by all tour vectors. This approach has been the most successful exact solution procedure proposed to date for the TSP; surveys of the large body of literature on the approach can be found in Jünger, Reinelt, and Rinaldi [12] and Naddef [14].

For any $S \subseteq V$, let $\delta(S)$ denote the set of edges with exactly one end in S and let $E(S)$ denote the set of edges having both ends in S. For disjoint sets $S, T \subseteq V$, let $E(S : T)$ denote the set of edges having one end in S and one end in T. For any set $F \subseteq E$, define $x(F) := \sum(x_e : e \in F)$.

Every tour of G satisfies the *subtour constraints* $x(\delta(S)) \geq 2 \quad \forall\, \emptyset \neq S \subsetneq V$. An important property of these constraints is that the corresponding separa-

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 452–467, 2005.

tion problem can be solved efficiently, that is, given a non-negative vector x^* a violated constraint can be found in polynomial time, provided one exists.

Much of the TSP literature is devoted to the study of classes of inequalities that are valid for the TSP, extending the subtour constraints in different ways. Many properties of these classes of inequalities are known, but for the most part polynomial-time separation algorithms have proven to be elusive. A notable exception is the separation algorithm for blossom-inequalities by Padberg and Rao [17]; variations of the Padberg-Rao algorithm are included in most current codes for the TSP. The absence of other efficient separation algorithms has lead to the use of various heuristic methods for handling TSP inequalities within cutting-plane algorithms. The heuristics are effective in many cases (see Padberg and Rinaldi [18], Applegate et al. [1], and Naddef and Thienel [16]), but additional exact methods could be critical in pushing TSP codes on to larger test instances.

An interesting new approach to TSP separation problems was adopted by Letchford [13], building on earlier work of Fleischer and Tardos [8]. Given an LP solution vector x^*, the support graph G^* is the subgraph of G induced by the edge-set $E^* = \{e \in E : x^*_e > 0\}$. Letchford [13] introduced a new class of TSP inequalities, called domino-parity constraints, and provided a separation algorithm in the case where G^* is a planar graph. An initial computational study of this algorithm by Boyd et al. [4], combining a computer implementation with by-hand computations, showed that the method can produce strong cutting planes for instances with up to 1,000 nodes.

In this paper we present a further study of Letchford's algorithm. We begin by describing a generalization of domino-parity inequalities and Letchford's algorithm to include certain multi-handled configurations. We also include a range of procedures for improving the practical performance of the separation routines, together with computational testing of large TSPLIB instances.

2 The k-Parity Inequalities

Definition 1. *Consider a family of sets $(T_1, T_2, \ldots, T_k; T)$ satisfying $\emptyset \neq T_i \subsetneq T \subsetneq V$, $\forall i \in I_k \equiv \{1, \ldots, k\}$. We call this family a* regular k-domino *if for any set $\emptyset \neq K \subseteq I_k$, the edges $\bigcup \{E(T_i : T \setminus T_i) : i \in K\}$ define a $|K|+1$ (or greater) cut in the subgraph of G induced by T. The family is called a* degenerate k-domino *if $(T_1, \ldots, T_k)$ defines a partition of T. We refer to the $k-$domino $(T_1, \ldots, T_k; T)$ as T and define β_T as 1 if T is regular and as $\frac{k}{k-1}$ if T is degenerate. In general, we say that the sets $T_1, \ldots, T_k$ are the* halves *of T. Finally, if T is a k-domino, we say that $\kappa(T) = k$.*

Lemma 1. *Let $T = (T_1, T_2, \ldots, T_k; T)$ be a $k-$domino. If x satisfies all subtour constraints, then*

$$\frac{\beta_T}{2}(x(\delta(T)) - 2) + \sum_{i=1}^{k} x(E(T_i : T \setminus T_i)) \geq k.$$

Proof. Assume x satisfies all subtour constraints. Let $B_1, B_2, \ldots, B_r$ correspond to the partition of T obtained by removing the edge sets $E(T_1 : T \setminus T_1), E(T_2 : T \setminus T_2), \ldots, E(T_k : T \setminus T_k)$. Then

$$\sum_{i=1}^{r} x(\delta(B_i)) = x(\delta(T)) + \sum_{i=1}^{r} x(E(B_i : T \setminus B_i)).$$

It follows that

$$\frac{\beta_T}{2}\left(x(\delta(T)) - 2\right) = \frac{\beta_T}{2}\left(\sum_{i=1}^{r}\left(x(\delta(B_i)) - x(E(B_i : T \setminus B_i))\right) - 2\right). \quad (1)$$

However, note that if T is regular, then $\beta_T = 1$ and

$$\sum_{i=1}^{r} x(E(B_i : T \setminus B_i)) \leq 2\sum_{i=1}^{k} x(E(T_i : T \setminus T_i)).$$

On the other hand, if T is degenerate, then $\beta_T \leq 2$ and each T_i can be assumed equal to B_i. Thus, in either case we have

$$\frac{\beta_T}{2}\sum_{i=1}^{r} x(E(B_i : T \setminus B_i)) \leq \sum_{i=1}^{k} x(E(T_i : T \setminus T_i)). \quad (2)$$

Finally, note that if T is regular, then $r > k$ and $\beta_T = 1$. Likewise, if T is degenerate then $r = k$ and $\beta_T = k/(k-1)$. Thus, in either case, $\beta_T(2r-2)/2 \geq k$, and

$$\frac{\beta_T}{2}\left(\sum_{i=1}^{r} x(\delta(B_i)) - 2\right) \geq \frac{\beta_T}{2}(2r-2) \geq k \quad (3)$$

Putting together (1), (2), and (3) we get the desired result. □

Definition 2. *Consider a family of teeth $\mathcal{T}$ and a family of handles $\mathcal{H}$, where each $T \in \mathcal{T}$ is a $\kappa(T)$-domino (with $\kappa(T) \leq |\mathcal{H}|$)and each $H \in \mathcal{H}$ is a proper subset of V. We say that Λ defines a* proper tooth-handle relationship *on $\mathcal{T}$ and $\mathcal{H}$ if we have the following (symmetric) associations. Each tooth $T \in \mathcal{T}$ is associated with exactly $\kappa(T)$ handles $H \in \mathcal{H}$, call this set $\Lambda(T)$, and each handle $H \in \mathcal{H}$ is associated with an odd number of dominoes $T \in \mathcal{T}$, call this set $\Lambda(H)$. For ease of notation, we index the halves of T according to the handle to which they are associated, that is, the halves of T are labeled $\{T_H\}_{H \in \Lambda(T)}$.*

Definition 3. *Let $\mathcal{F} = \{E_1, E_2, \ldots, E_k\}$, where $E_i \subseteq E$ for all $i \in I_k$, and define $\mu_e := |\{F \in \mathcal{F} : e \in F\}|$ for each $e \in E$. Following Letchford [13], the family $\mathcal{F}$ is said to* support *the cut $\delta(H)$ if $\delta(H) = \{e \in E : \mu_e$ is odd$\}$.*

Theorem 1. *Suppose that Λ defines a proper tooth-handle relationship on $\mathcal{T}$ and $\mathcal{H}$. For each $H \in \mathcal{H}$ let $F_H \subseteq E$ be such that $\{F_H, \{E(T_H : T \setminus T_H)\}_{T \in \Lambda(H)}\}$ supports the cut $\delta(H)$ in G and define μ^H accordingly. Then the inequality*

$$\sum_{H \in \mathcal{H}} \mu^H x + \sum_{T \in \mathcal{T}} \beta_T x(\delta(T)) \geq \sum_{H \in \mathcal{H}} |\Lambda(H)| + 2 \sum_{T \in \mathcal{T}} \beta_T + |\mathcal{H}| \tag{4}$$

is satisfied by all tours.

Proof. We use induction on $|\mathcal{H}|$, the case $|\mathcal{H}| = 0$ following from the validity of the subtour constraints. Let x^c be the incidence vector of a tour. If there exists $H_o \in \mathcal{H}$ such that $\mu^{H_o} x^c > |\Lambda(H_o)| - 1$, then, since $\mu^{H_o} x^c$ is even valued (see Letchford [13]), we have $\mu^{H_o} x^c \geq |\Lambda(H_o)| + 1$. Note also that for each $T \in \Lambda(H_o)$ the family $\{T_H : H \in \Lambda(T) \setminus H_o; T\}$ defines a regular $(|\Lambda(T)| - 1)-$domino. Thus, by induction, the inequality obtained by removing H_o and redefining $\beta'_T = \beta_T$ for $T \in \mathcal{T} \setminus \Lambda(H_o)$ and $\beta'_T = 1$ for $T \in \Lambda(H_o)$

$$\sum_{H \in \mathcal{H} \setminus H_o} \mu^H x + \sum_{T \in \mathcal{T}} \beta'_T x(\delta(T)) \geq \sum_{H \in \mathcal{H} \setminus H_o} |\Lambda(H)| + (|\mathcal{H}| - 1) + 2 \sum_{T \in \mathcal{T}} \beta'_T$$

is valid. Then (4) follows since $(\beta_T - \beta'_T) x^c(\delta(T)) \geq (\beta_T - \beta'_T) 2$, and $\mu^{H_o} x^c \geq |\Lambda(H_o)| + 1$.

So we can now assume that $\mu^H x^c \leq |\Lambda(H)| - 1$ for each $H \in \mathcal{H}$. From Lemma 1 we have for each $T \in \mathcal{T}$

$$\beta_T (x^c(\delta(T)) - 2) \geq 2|\Lambda(T)| - 2 \sum_{H \in \Lambda(T)} x^c(E(T_H : T \setminus T_H)).$$

Hence,

$$\begin{aligned}
\sum_{T \in \mathcal{T}} \beta_T (x^c(\delta(T)) - 2) &\geq 2 \sum_{T \in \mathcal{T}} |\Lambda(T)| - 2 \sum_{T \in \mathcal{T}} \sum_{H \in \Lambda(T)} x^c(E(T_H : T \setminus T_H)) \\
&\geq 2 \sum_{H \in \mathcal{H}} |\Lambda(H)| - 2 \sum_{H \in \mathcal{H}} \mu^H x^c \\
&= \sum_{H \in \mathcal{H}} |\Lambda(H)| + \sum_{H \in \mathcal{H}} \left(|\Lambda(H)| - \mu^H x^c \right) - \sum_{H \in \mathcal{H}} \mu^H x^c \\
&\geq \sum_{H \in \mathcal{H}} |\Lambda(H)| + |\mathcal{H}| - \sum_{H \in \mathcal{H}} \mu^H x^c.
\end{aligned}$$

□

We refer to the constraints (4) as *k-parity inequalities*, when $|\mathcal{H}| = k$. When $k = 1$ this class is precisely the *domino-parity inequalities* of Letchford [13]. It is easy to see that not all k-parity inequalities define facets of the TSP polytope, but the class does provide a common framework for possibly extending Letchford's algorithm to superclasses of other inequalities that have proven to be effective in TSP codes. In particular, k-parity inequalities generalize clique-tree inequalities (Grötschel and Pulleyblank [10]) in the same way as domino-parity inequalities generalize combs.

Definition 4. *Families $\mathcal{H}$ and $\mathcal{T}$ are said to define a* clique-tree *if:*

(i) $\mathcal{H}$ is a family of pairwise disjoint proper subsets of V.
(ii) $\mathcal{T}$ is a family of pairwise disjoint proper subsets of V.
(iii) No $T \in \mathcal{T}$ is contained in $\bigcup(H : H \in \mathcal{H})$.
(iv) For each $H \in \mathcal{H}$ let $\Lambda(H) = \{T : T \cap H \neq \emptyset\}$. $|\Lambda(H)|$ must be odd.
(v) The intersection graph defined by the families $\mathcal{H}$ and $\mathcal{T}$ is a tree.

In this context, the sets $H \in \mathcal{H}$ are called handles and the sets $T \in \mathcal{T}$ are called teeth. If the intersection graph defined by the families $\mathcal{H}$ and $\mathcal{T}$ is a forest, we say that $\mathcal{H}$ and $\mathcal{T}$ define a clique-forest. Note that if $\mathcal{H}$ and $\mathcal{T}$ define a clique-forest, then it is possible to define a $|\mathcal{H}|$-parity constraint as follows. For each $T \in \mathcal{T}$ define $\Lambda(T) = \{H \in \mathcal{H} : T \in \Lambda(H)\}$, and $T_H = T \cap H$, $\forall H \in \Lambda(T)$. Clearly $(T_H : H \in \Lambda(T); T)$ defines a $|\Lambda(T)|$-domino and $(\mathcal{H}, \mathcal{T}, \Lambda)$ defines a proper tooth-handle relationship. Thus, Theorem 1 implies that the well-known clique-tree (forest) constraint is valid $\sum_{H \in \mathcal{H}} x(\delta(H)) + \sum_{T \in \mathcal{T}} x(\delta(T)) \geq 2|\mathcal{T}| + |\mathcal{H}| + \sum_{H \in \mathcal{H}} |\Lambda(H)|$ where $\sum(|\Lambda(H)| : H \in \mathcal{H})$ is commonly written as $|\mathcal{T}| + |\mathcal{H}| - 1$. Clique-tree inequalities generalize combs inequalities, which are clique trees having a single handle.

We will focus on special cases of clique trees in the next section, but we would like to point out that k-parity inequalities also generalize several other well-known classes of TSP constraints.

Proposition 1. *The family of $k-$parity inequalities generalizes the family of bipartition inequalities and the family of star inequalities.*

3 Planar Separation with Multiple Handles

Throughout this section we assume that the LP solution x^* satisfies all subtour constraints. Also, for any set $F \subseteq E$, we define $F^* = \{e \in F : x^*_e > 0\}$.

Definition 5. *For a given $x^* \in SEP(n)$, We say that a k-domino $(T_1, \ldots, T_k; T)$ is* super-connected *if:*

(i) T and $V \setminus T$ are connected in G^*.
(ii) T_i and $T \setminus T_i$ are connected in G^* for all $i \in I_k$.
(iii) $x^*(E(T_i : V \setminus T)) > 0$ and $x^*(E(T \setminus T_i : V \setminus T)) > 0$ for all $i \in I_k$.
We say that a k-parity constraint having teeth $\mathcal{T}$ is super-connected, if every tooth $T \in \mathcal{T}$ is super-connected.

While as of yet it is an open problem whether or not the class of k-parity inequalities can be separated in polynomial time, we extend the ideas of Letchford [13] so as to separate, for fixed k, a subclass of k-parity inequalities which contains all super-connected clique-trees with k handles or less, under the assumption that the support graph G^* is planar.

For this we proceed in three steps. First, we characterize violated k-parity inequalities. Second, we characterize violated k-parity inequalities under the additional assumptions that the support graph G^* is planar, and that teeth are

super-connected. Finally, we outline an algorithm for separating a subclass of k-parity inequalities when G^* is planar; this subclass (defined with respect to an LP solution x^*) contains all super-connected clique-tree inequalities which have k handles or less.

The following two Propositions (the proofs of which are to be included in a future paper) are not used throughout the following sections. However, they serve as a motivation for separating classes of super-connected constraints.

Proposition 2. *Let x^* be an LP solution and consider a violated clique-tree constraint on k handles, having teeth $\mathcal{T}$. Let $(T_1, T_2, \ldots, T_q; T) \in \mathcal{T}$. If all clique-tree constraints having less than k handles are satisfied by x^*, then (a) T is connected, (b) T_i is connected for all $i \in I_q$, (c) $x^*(E(T_i : V \setminus T)) > 0$ and $x^*(E(T \setminus T_i : V \setminus T)) > 0$ for all $i \in I_q$.*

Proposition 3. *If all subtour inequalities are satisfied, then there exists a maximally violated (if any) comb inequality which is super-connected. If all subtour and comb inequalities are satisfied, then there exists a maximally violated (if any) clique-tree inequality on two handles which is super-connected.*

Proposition 2 indicates that when clique-tree inequalities on k handles are satisfied, then all violated clique-trees on $k+1$ handles are *almost* super-connected. Proposition 3 shows that once comb inequalities are effectively separated, we may assume for exact separation purposes that two handled clique-trees are super-connected.

3.1 Characterizations of Violated k-Parity Constraints

Definition 6. *Define the* weight *of k-domino $(T_1, T_2, \ldots, T_k; T)$ to be $w(T) := \beta_T(x(\delta(T)) - 2) + \sum_{i=1}^{k} x(E(T_i : T \setminus T_i)) - k$.*

Lemma 2. *The slack of a k-parity inequality is $\sum_{T \in \mathcal{T}} w(T) + \sum_{H \in \mathcal{H}} x(F_H) - |\mathcal{H}|$.*

Note that Lemma 1 and Lemma 2 together imply that a violated k-parity constraint must satisfy

$$0 \leq \frac{\beta_T}{2}(x(\delta(T)) - 2) \leq w(T) \leq |\mathcal{H}| \qquad \forall T \in \mathcal{T}. \tag{5}$$

Definition 7. *Consider a family of teeth $\mathcal{T}$, where each $T \in \mathcal{T}$ satisfy $\kappa(T) \leq k$. We say that Φ defines an* abstract tooth-handle *relationship over $\mathcal{T}$ and I_k if (i) $\Phi(T) \subseteq I_k$ and $|\Phi(T)| = \kappa(T)$ for all $T \in \mathcal{T}$, (ii) $\Phi(i) \subseteq \mathcal{T}$ and $|\Phi(i)|$ is odd, for all $i \in I_k$, and (iii) $T \in \Phi(i)$ iff $i \in \Phi(T)$ for all $i \in I_k, T \in \mathcal{T}$.*

Lemma 3. *There exists a violated k-parity inequality iff there exist $(\mathcal{T}, \Phi)$ defining an abstract tooth-handle relationship, and sets $R_i \subseteq E^*$ for all $i \in I_k$ such that:*

(i) $\{E^*(T_i : T \setminus T_i)\}_{T \in \Phi(i)}$ *and* $\{R_i\}$ *support a cut in* G^* *for all* $i \in I_k$.
(ii) $\sum_{i \in I_k} x^*(R_i) + \sum_{T \in \mathcal{T}} w(T) - k < 0$.

Proof. From Theorem 1 and Lemma 2, a k-parity inequality is violated iff there exist $(\mathcal{H}, \mathcal{T}, \Lambda)$ defining a proper tooth-handle relationship, and sets $F_H \subseteq E$ for $H \in \mathcal{H}$ such that:

(a) $\{E(T_H : T \setminus T_H)\}_{T \in \Lambda(H)}$ and $\{F_H\}$ support the cut $\delta(H)$ in G for all $H \in \mathcal{H}$
(b) $\sum_{H \in \mathcal{H}} x^*(F_H) + \sum_{T \in \mathcal{T}} w(T) - |\mathcal{H}| < 0$

We first prove necessity. Assume that $(\mathcal{H}, \mathcal{T}, \Lambda)$ defines a violated k-parity inequality. We know that there exists $F_H \subseteq E$ for $H \in \mathcal{H}$ satisfying (a)-(b). Assume $\mathcal{H} = \{H_i : i \in I_k\}$. For each $i \in I_k$ define $\Phi(i) = \Lambda(H_i)$, $\Phi(T) = \{i : H_i \in \Lambda(T)\}$ and $R_i = F_{H_i} \cap E^*$. Note that Φ and $\mathcal{T}$ define an abstract tooth-handle relationship, and $|\mathcal{H}| = k$. Hence, conditions (a)-(b) imply (i)-(ii).

We next prove sufficiency. Assume that $\mathcal{T}, \Phi$ define an abstract tooth-handle relationship, and sets $R_i \subseteq E^*$, $i \in I_k$ are such that (i) and (ii) hold. For each $i \in I_k$ let $H_i \subseteq V$ be one shore of the cut supported by $\{E^*(T_i : T \setminus T_i)\}_{T \in \Phi(i)}$ and R_i, and let $\Lambda(H_i) = \Phi(i)$. Likewise, for $T \in \mathcal{T}$ define $\Lambda(T) = \{H_i : i \in \Phi(T)\}$. Note that Λ define a proper tooth-handle relationship on $\mathcal{T}, \mathcal{H}$. Define $F_{H_i} \subseteq R_i \cup \{e \in \delta(H_i) : x_e = 0\}$ such that (a) holds. Thus (b) must also hold. □

For the remainder of this section, assume that G^* is a planar graph and let $\bar{G}^*$ denote the planar dual of G^*. For any subset $F \subseteq E(G^*)$, denote by $\bar{F}$ the corresponding edges in $\bar{G}^*$. For each $\bar{e} \in \bar{G}^*$ let $x^*_{\bar{e}} = x^*_e$.

Definition 8. *A graph* H *is called* Eulerian *if every node has even degree. (As in Letchford [13], we do not require that* H *be connected.)*

Definition 9. *Let* r *be a positive integer and suppose that* $E_1, \ldots, E_r$ *are edge-sets satisfying* $E_i \subseteq E^*$, $i \in I_r$. *The collection* $\{\bar{E}_i : i \in I_r\}$ *is said to* support an Eulerian subgraph *in* $\bar{G}^*$ *if the edges* $\bar{e}$ *for which* μ_e *is odd form an Eulerian subgraph in* $\bar{G}^*$.

This definition implies that $\{\bar{E}_i : i \in I_r\}$ supports an Eulerian subgraph in $\bar{G}^*$ iff $\{E_i : i \in I_r\}$ supports a cut in G^*. Hence we have the the following dual version of Lemma 3.

Lemma 4. *A* k*-parity inequality is violated iff there exist* $\mathcal{T}, \Phi$ *defining an abstract tooth-handle relationship, and sets* $\bar{R}_i \subseteq \bar{E}^*$ *for* $i \in I_k$ *such that:*

(i) $\{\overline{E^*(T_i : T \setminus T_i)}\}_{T \in \Phi(i)}$ *and* $\{\bar{R}_i\}$ *support an Eulerian subgraph in* $\bar{G}^*$ *for all* $i \in I_k$.
(ii) $\sum_{i=1}^{k} x^*(\bar{R}_i) + \sum_{T \in \mathcal{T}} w(T) - k < 0$

Lemma 5. *A k-domino $(T_i : i \in I_k; T)$ is super-connected iff (a) $C(T) = \overline{\delta^*(T)}$ is a simple cycle in $\bar{G}^*$, (b) for each $i \in I_k$ the edges $P_i(T) = \overline{E^*(T_i : T \setminus T_i)}$ define a simple path in $\bar{G}^*$ with end-points $\{s_i^T, t_i^T\}$ in $C(T)$ (where $s_i^T \neq t_i^T$) and all other nodes not in $C(T)$, and (c) all of the paths $P_i(T)$ are in the same side of the cycle C with respect to the planar embedding.*

Definition 10. *Consider two distinct super-connected k-dominoes T and L. If the end-points of $P_i(T)$ and $P_i(L)$ are the same for $i \in I_k$ and $w(T) < w(L)$ we say that T* dominates *L.*

Lemma 6. *Consider two distinct super-connected k-dominoes T and L. If T dominates L, and if L is used in some violated k-parity constraint, then L may be replaced by T to obtain another violated k-parity constraint has less slack.*

Proof. By removing each path $P_i(L)$ and replacing it with $P_i(T)$ for $i \in I_k$, condition (i) of Lemma 4 is not changed, and $w(T) < w(L)$ implies condition (ii) is not changed - in fact, the violation, given by (ii), will improve from the substitution. □

From Lemma 6 it follows that a maximally violated super-connected k-parity constraint will only have non-dominated teeth.

3.2 Separating Super-Connected Clique Tree Constraints

Given a fixed $k \in \mathbb{Z}_+$ and a fractional LP solution x^* satisfying all subtour constraints, the algorithm proceeds in two steps. First, a minimal family of non-dominated teeth is generated. Next, a violated super-connected k-handle constraint is generated (if such exists) by solving an odd Eulerian subgraph problem in an appropriate graph.

In order to describe the tooth generation procedure, it is important to establish two results.

Lemma 7. *Every tooth T in a violated k-handle clique-tree constraint must satisfy $2 \leq x^*(\delta(T)) < 2(k+1)$.*

Proof. Follows from (5), the subtour constraints, and the fact that clique-trees have no degenerate teeth. □

Lemma 8. *Consider a violated super-connected k-handled clique-tree constraint with tooth set $\mathcal{T}$. Let $(T_i : i \in I_{\kappa(T)}; T)$ be a $\kappa(T)$-domino in $\mathcal{T}$, and define P_i to be the path $\overline{E(T_i : T \setminus T_i)}$ for all $i \in I_{\kappa(T)}$. Then,*

(i) Paths P_i and P_j don't cross with regards to the dual embedding, for $i \neq j \in I_{\kappa(T)}$.

(ii) Paths P_i and P_j can't have the same end-points, unless $\kappa(T) = 2$ for $i \neq j \in I_{\kappa(T)}$.

(iii) If paths P_i and P_j have the same end-points, then $P_i \neq P_j$, for $i \neq j \in I_{\kappa(T)}$.

Proof. For (i) If paths P_i and P_j cross, then halves T_i and T_j must intersect. For (ii) assume that P_i and P_j have the same end-points. If there exists a path P_k with $k \neq i, j$, given that it can't intersect paths P_i or P_j, it must either run between P_i and P_j, or must run the side of either P_i or P_j. In either case, this implies that T_k intersects T_i or T_j. For (iii) if P_i and P_j have the same end-points and the paths coincide, then the tooth must be degenerate. However, this is contradictory with the definition of clique-trees. □

Lemma 7 and Lemma 8 suggest a natural algorithm by which to enumerate a minimal set of teeth for a violated super-connected k handle clique-tree constraint. First, enumerate all connected sets $T \subseteq V$ which satisfy the condition in Lemma 7 using an algorithm such as that of Nagamochi et. al [15]. Keep those sets T for which $V \setminus T$ is connected. Let $C = \overline{\delta(T)}$. Choose a side of C with regard to the planar embedding, and let W represent the nodes of that side minus the nodes in C. Next, for each pair of nodes $u, v \in C$ compute the shortest path and second-shortest path from u to v in W. Choose $q \in I_k$ and a set of end-points $\{(s_i, t_i) : i \in I_q\}$ in C. Check that that no two pairs of end-points are crossing (that is, such that it is impossible to take a path from s_i to t_i without crossing a path from s_j to t_j). If $q = 2$ and $s_1 = s_2 = s$, $t_1 = t_2 = t$, let P_1 be the shortest s to t path, and let P_2 be the second-shortest s to t path. Otherwise, define $\hat{P}_i$ as the shortest s_i to t_i path for $i \in I_q$. If the paths $\hat{P}_1, \ldots, \hat{P}_q$ cross each other, un-cross them so as to define paths P_i, $i \in I_q$. At this stage, C and the paths P_i, $i \in I_q$, define a q-domino. If the weight is larger than k, or, if there is another q-domino which dominates it, discard the tooth. Keep iterating until all possible combinations of end-points, sides of the cycle, and sets T have been exhausted. It is not difficult to see that this algorithm is polynomial, and that it enumerates a minimal set of non-dominated teeth (which is polynomially sized).

For the specific case in which $k = 1$, a faster tooth generation procedure is presented in Letchford [13]. First, if $k = 1$, it is shown that a tooth T is super-connected iff $\overline{\delta(T)}$ and $\overline{E(T_1 : T \setminus T_1)}$ define three node-disjoint paths in $\bar{G}^*$. In order to construct the teeth, a network $\mathcal{N}$ is constructed from the graph $\bar{G}^*$ so that the nodes of $\mathcal{N}$ and $\bar{G}^*$ coincide. Then, for each edge in $\bar{G}^*$, two arcs (one in each direction) of capacity one are added to $\mathcal{N}$. By solving the min-cost three-unit flow problem between each pair of nodes in $\mathcal{N}$, it is possible to generate a minimal set of non-dominated teeth. The fact that paths in a solution may possibly cross is not a problem, for it is shown that if an optimal solution is crossing, then for the given pair of nodes there can be no tooth satisfying condition (5). In our implementation of Letchford's algorithm we use this idea, which will be further discussed in Section 4. This brings us to our main result.

Theorem 2. *Suppose G^* is planar and x^* satisfies all subtour constraints. Consider a fixed integer $k \geq 1$. It is possible to separate in polynomial time a subclass of k-parity constraints which contains all violated super-connected clique-tree inequalities on k handles.*

The sketch of the proof of this theorem consists of two parts. First we outline a two-stage algorithm which runs in polynomial time, and then we enunciate an algorithm that separates a subclass of k-parity constraints which contains all violated super-connected clique-tree inequalities.

The two steps of the algorithm are as follows:

(i) Construct a minimal non-dominated family of teeth $\mathcal{L}$.
(ii) Construct a graph $M[k]$ using $\mathcal{L}$ and $\bar{G}^*$. Solve the min-weight 1^k-Eulerian Subgraph problem in $M[k]$.

Given $G = (V, E)$, edge weights $w : E \to \mathbb{R}_+$, and edge parities $p : E \to \{0, 1\}^k$, the min-weight 1^k-Eulerian Subgraph problem asks for an Eulerian subgraph M^* of G of minimum weight with parity $p(M^*) := \sum(p(e) : e \in M^*) = 1^k \mod 2$. If we obtain a solution in step (ii) having weight less than k, then we have found a violated k-parity constraint. The intuition of the algorithm is as follows: From Lemma 4 we know that a violated k-parity inequality can be characterized by a set of Eulerian-subgraphs of G^*, one for each handle, and each utilizing an odd number of teeth. For every path $P_i(T) = \overline{E^*(T_i : T \setminus T_i)}$ define an *odd* edge whose end-points coincide with the end-points of $P_i(T)$, and whose weight coincides with the weight of the tooth. Thus, the problem can be modeled as that of searching for a set of odd Eulerian subgraphs (those that use an odd number of odd edges), one for each handle, whose combined weight is minimized, subject to side constraints (defined by the teeth) which link these subgraphs to each other. The side constraints would impose that either all of the paths associated to a tooth are used (in different handles), or none at all. The proposed algorithm works by defining a graph $M[k]$ which contains the Cartesian product of k copies of $\bar{G}^*$; the idea being that any path in $M[k]$ corresponds to k individual paths in $\bar{G}^*$, one in each of the components (or layers) which make up the Cartesian product. By defining special edges in $M[k]$ associated to teeth in $\mathcal{L}$, it is possible to associate certain Eulerian subgraphs in $M[k]$ to k-parity inequalities defined in $\bar{G}^*$. Note that $M[1]$ coincides with the graph M^* as defined in Letchford [13]; in this case Letchford proved that the condition of being Eulerian can be replaced by the condition of being a simple cycle.

4 Implementation and Computational Results

In this section we briefly describe our computational tests. First we discuss the implementation of Letchford's algorithm for separating domino-parity constraints, emphasizing the techniques we adopted to improve its practical performance, and presenting some computational results. Next, we discuss the implementation of a simple heuristic for separating 2-parity constraints.

4.1 Domino-Parity Constraints

Domino Searching. Teeth were generated by using the network flow approach described in Section 3.2 and Letchford [13]. To find the min-weight node-disjoint

paths between pairs $s, t \in V(\bar{G}^*)$ we used the augmenting-shortest-path network-flow algorithm (See Ahuja et al [3] for details). For this, we build a network $\mathcal{N}$ for the graph $\bar{G}^*$ defining two arcs for each edge (one in each direction), assigning a capacity of one to each. This algorithm computes the $s-t$ flow by solving three successive $s-t$ shortest path problems on reduced capacity networks successively derived from $\mathcal{N}$. Using this algorithm several speed-ups were possible. Firstly, for fixed $s \in V(\bar{G}^*)$ the first $s-t$ flow for all nodes t can be obtained by solving a single Dijkstra algorithm in $\mathcal{N}$ rooted at s. The additional shortest-path computations need only be computed for nodes t at distance not greater than 4/3 from s. Finally, when computing the $s-t$ three-flow in $\mathcal{N}$, one only need consider intermediary nodes at distance not greater than 2 from s and t. This follows from the fact that every cycle in $\bar{G}^*$ corresponds to a cut in G^*, and hence, has weight at least 2 (due to subtour constraints). Thus, if a node is used which has distance at least 2, since the other two paths will define a cycle, the bound of 4 would be exceeded. A useful heuristic idea is to further restrict the set of intermediary nodes to those of distance no greater than 2α for some $\alpha < 1$; this restriction can cause the algorithm to miss violated DP-cuts, but it greatly improves the speed and appears to work well in practice (we have set $\alpha = .55$ in our tests).

Parallelization. Dominoes may be computed in parallel. In fact, one may divide the nodes $s \in V(\bar{G}^*)$ among different machines so that each one computes all of the (s,t) three-disjoint paths. We found the domino-computation stage to be (by far) the most time consuming part of the algorithm, making this parallelization crucial for obtaining acceptable running times on large instances when using $\alpha = 1$. Our parallel implementation is a master–worker system based on message passing.

Random Walk. The algorithm as formally defined in Letchford [13] computes exactly one constraint. In practice, one would like the algorithm to compute as many violated constraints as possible. To achieve this, instead of just solving the shortest odd-cycle problem in M^* we additionally run a random walk algorithm that attempts to find small-weight odd cycles. This algorithm is fast, easy to implement, and in our tests generally produced a large number of additional cuts, only the best of which were kept.

Safe Shrinking. The size of the graph G^* has a dramatic impact on the running time of our implementation. Following the work of Padberg and Rinaldi [18], we attempt to reduce the size of G^* by contracting edges in G^*, redefining the vector x^*, and solving the separation problem in the new, smaller, graph. In this *shrinking* process, a contraction is called *safe* if we know that the existence of a violated DP-inequality implies the existence of one in the graph we obtain after the contraction. Although it is not always the case that shrinking is safe, it is possible to give conditions under which it will be.

Theorem 3. *Consider x^* satisfying all subtour constraints, a DP-inequality $ax \leq b$ satisfying $ax^* > b$, and nodes $u, v, t \in V(G^*)$ such that $x^*_{uv} = 1$, and*

$x^*_{ut} + x^*_{vt} = 1$. *If* $a_{u,v} \neq 0$ *there exists another DP-inequality* $a'x \leq b'$ *such that* $a'_{u,v} = 0$ *and* $(a'x^* - b') \geq (ax^* - b)$. *Thus, we can contract edge* $\{u, v\}$ *and ensure the existence of a maximally violated DP inequality with zero* $\{u, v\}$ *coefficient.*

As a pre-processor to our implementation, we repeatedly contract edges $\{u, v\}$ while there exist nodes u, v, t satisfying the conditions of Theorem 3.

Planarity. The safe-shrinking procedure can greatly reduce the size of the graph over which we work, but if the original graph G^* is non-planar then the shrunk may too be non-planar. If this is the case, our implementation does non-safe shrinks until a planar graph is obtained, as in Boyd et al. [4]. If G^* is not planar, we identify a forbidden $K_{3,3}$ or K_5 minor $M \subseteq E(G^*)$. We then take two nodes in the minor with degree at least 3 and contract them (and thus eliminating the minor), iterating until a planar graph is obtained. An alternative is to eliminate an edge $e \in M$ from G^*, iterating until a planar graph is obtained. There are several ways in which M and $e \in M$ may be selected, and we found that the way in which the selection is made can make an important difference in the performance of the algorithm.

Tightening. After adding a cutting plane to an LP and re-solving, it is possible that we may obtain another fractional solution that differs very little from the one just separated. In this case, rather than generating new cuts all over again, it may be desirable to attempt to "fix up" some tight constraints currently in the LP or in the cut-pool by slightly modifying them in such a way as to make the new fractional point infeasible (or make an already violated constraint more violated). This is certainly much faster than separating from scratch, and also does not require G^* to be planar. This type of approach has been very successful on other classes of inequalities (see Applegate et al. [2]) and it had a great impact in our computational results.

To formalize this notion of *simple modifications* for DP-inequalities, recall that every DP-inequality is completely defined by a family of dominoes $\{A_i, B_i\}_{i=1}^k$ and a handle H. Thus, adding and/or deleting a node from any of those sets will result in slight changes of the constraint which potentially could result in a new, violated cut.

In our implementation we consider the following set of simple modifications. Given a node in G^*, we can (i) add it/remove it from a domino; (ii) have it switch sides in a domino; (iii) add it/remove it from the handle; (iv) do some combinations of the previous modifications. We implemented a greedy heuristic which computes the best move for every relevant node[1], and while the best move (among all nodes) reduces the slack of the constraint, perform the move, and update the best move for the relevant nodes in the graph. If all remaining best moves are zero-valued (that is, they do not change the slack), we first do moves that enlarge either the handle or a domino, then do moves that flip elements

[1] A node is relevant in the heuristic if $\exists e \in$ ( ) such that it has a non-zero coefficient in the DP-inequality.

within a domino and then do moves that shrink a domino or a handle. We repeat this until some improving move is found or until we cannot make any more moves.

TSPLIB Tests. In Tables 1, 2, and 3 we report on a set of tests on all instances from the TSPLIB having at least 3,000 cities. The computations were performed on a single processor of a dual 2.66 GHz Intel Xeon Linux workstation. The LP solver used was ILOG CPLEX 6.5. The algorithm used for planarity testing was Boyer and Myrvold[2] [5].

Table 1. DP-Cuts on TSPLIB Instances

Name	Optimal	Concorde	DP	Gap	Concorde Hours	DP Hours
pcb3038	137694	137660	137687	79%	24.9	8.6
fl3795	28772	28697	28772	100%	21.2	8.2
fnl4461	182566	182555	182559	36%	7.9	3.4
rl5915	565530	565384	565482	67%	103.7	46.1
rl5934	556045	555929	556007	67%	17.5	48.3
pla7396	23260728	23255280	23259532	78%	133.7	106.9

Table 2. DP-Cuts on Larger TSPLIB Instances

Name	Optimal	Concorde	Concorde+DP	Gap	Concorde Hours	DP Hours
usa13509	19982859	19979209	19981173	54%	81.2	72.0
brd14051	469385	469321	469352	48%	53.2	72.0
d15112	1573084	1572853	1572956	45%	114.0	72.0
d18512	(645238)	645166	645193	38%	74.0	72.0

Table 3. DP-Cuts on Largest TSPLIB Instances

Name	Optimal	Concorde (with pool)	Concorde+DP (with pool)	Gap
pla33810	66048945	66018619	66037858	63%
pla85900	(142382671)	142336550	142354693	39%

In the tests in Tables 1 and 2, we used the Concorde command line option -mC48 to allow Concorde to repeatedly call the local-cuts routine up to size 48 (see Applegate et al. [2]); this setting requires additional CPU time over the default version of Concorde, but it allows Concorde to obtain substantially better lower bounds. For each instance in Table 2 we also ran Concorde together with the DP-cut code starting from Concorde's final LP (cutting off the runs after 72 hours), while for each instance in Table 1 we ran Concorde with the DP-cut code starting from scratch.

[2] We give special thanks for J.M. Boyer for allowing us to use his implementation of the planarity testing algorithm.

In Table 3 we consider the two largest examples in the TSPLIB. Rather than working from scratch on these instances, we study the effectiveness of DP-cuts in improving the best available LP relaxations. In each instance, we begin with an LP found by Applegate et al. by gathering cuts into a pool during a sequence of 3 branch-and-cut runs (stopping each run after it reached 1,000 active subproblems). The LP was then improved by applying DP-cuts, with new cut pools gathered using 2 branch-and-cut runs for pla33810 and a single short run (to 75 active subproblems) for pla85900.

As a case-study, starting from the 66,037,858 LP, we have established the optimal value of 66,048,945 for pla33810. The optimal tour is a slight improvement on the best reported tour of value 66,050,499, found by Helsgaun [11] with a variant of his LKH heuristic. The branch-and-cut run that solved the instance used 577 subproblems (given the upper bound of 1 larger than Helsgaun's LKH tour). We also solved the instance a second time starting with a 66,037,858 LP (obtained using the cuts from the earlier run) and an upper bound of 1 greater than the optimal value; the branch-and-cut run in this case used 135 subproblems.

Our solution of pla33810 should be viewed only as evidence of the potential strength of the new procedures; the computational study was made as we were developing our code and the runs were subject to arbitrary decisions to terminate tests as the code improved. The total CPU time used in the solution of pla33810 was approximately 15.7 CPU years (the additional branch-and-cut run of 135 nodes took 86.6 days).

The two remaining open problems in the TSPLIB are d18512 and pla85900; the tour values reported in our tables for these instances were obtained by Tamaki [19] and Helsgaun [11], respectively.

4.2 2-Parity Constraints

To test the efficacy of 2-parity constraints, we developed a heuristic that works by taking tight (or almost tight) domino-parity constraints, and attempts to grow a second handle. For this, consider a super-connected domino-parity constraint with teeth $\mathcal{T}$. The heuristic works in two stages. First, for every tooth $T \in \mathcal{T}$ shortest paths are computed between pairs of nodes in $\delta(T)$. Then, in a second stage, the algorithm attempts to connect an odd number of these paths into a simple cycle with edges in $\bar{G}^*$, by using a random-walk which gives preference to edges having small weight (with regard to the values given by the fractional vector x) and which forbids taking two different paths associated to a same tooth $T \in \mathcal{T}$. It is not difficult to see from Lemma 3 that such a structure in $\bar{G}^*$ corresponds to a 2-parity cut.

In Table 4 we report results using an implementation of the 2-parity separation heuristic on a selection of small TSPLIB instances that are not easily solvable at the root node. The "DP" column gives the LP value using Concorde with DP-cuts and with local cuts of size 32; the "DP+2P" column reports the LP value obtained by starting with the "DP" LP and running Concorde with the 2P-cut separator. The improvement in the LP gap varied widely, but it is promising that 2P-cuts can often strengthen these already very good LP bounds.

Table 4. 2-Parity Cuts on TSPLIB Instances

Name	Optimal	DP	DP+2P	Gap
pcb442	50778	50765	50765	0%
att532	27686	27685	27686	100%
dsj1000	18660188	18659299	18660093	89%
u1060	224094	224044	224054	20%
vm1084	239297	239294	239297	100%

References

1. Applegate, D., R. Bixby, V. Chvátal, W. Cook. 1998. On the solution of traveling salesman problems. Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung, International Congress of Mathematicians. 645–656.
2. Applegate, D., R. Bixby, V. Chvátal, W. Cook. 2003. Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems. Mathematical Programming **97**, 91–153.
3. Ahuja R.K., T.L. Magnanti, J.B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, New Jersey.
4. Boyd, S., S. Cockburn, D. Vella. 2001. On the domino-parity inequalities for the STSP. Computer Science Technical Report TR-2001-10. University of Ottawa.
5. Boyer, J. M., W. Myrvold. 2004. On the cutting edge: simplified O(n) planarity by edge addition. Journal of Graph Algorithms and Applications. To appear.
6. Chvátal, V. 1973. Edmonds polytopes and weakly hamiltonian graphs. Mathematical Programming **5**, 29–40.
7. Dantzig, G., R. Fulkerson, S. Johnson. 1954. Solution of a large-scale traveling salesman problem. Operations Research **2**, 393–410.
8. Fleischer, L., É. Tardos. 1999. Separating maximally violated comb inequalities in planar graphs. Mathematics of Operations Research **24**, 130–148.
9. Grötschel, M., O. Holland. 1991. Solution of large-scale symmetric travelling salesman problems. Mathematical Programming **51**, 141–202.
10. Grötschel, M., W. R. Pulleyblank. 1986. Clique tree inequalities and the symmetric travelling salesman problem. Mathematics of Operations Research **11**, 537–569.
11. Helsgaun, K. 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research **126**, 106–130.
12. Jünger, M., G. Reinelt, G. Rinaldi. 1995. The traveling salesman problem. M. Ball, T. Magnanti, C. L. Monma, G. Nemhauser, eds. *Handbooks on Operations Research and Management Sciences: Networks*. North Holland, Amsterdam, The Netherlands. 225–330.
13. Letchford, A. N. 2000. Separating a superclass of comb inequalities in planar graphs. Mathematics of Operations Research **25**, 443–454.
14. Naddef, D. 2002. Polyhedral theory and branch-and-cut algorithms for the symmetric traveling salesman problem. G. Gutin, A. Punnen, eds. *The Traveling Salesman Problem and Its Variations*. Kluwer, Dordrecht, pp. 29–116.
15. Nagamochi, H., K. Nishimura, T. Ibaraki. 1997. Computing all small cuts in undirected networks. SIAM Journal on Discrete Mathematics **10**, 469–481.
16. Naddef, D., S. Thienel. 2002. Efficient separation routines for the symmetric traveling salesman problem II: separating multi handle inequalities. Mathematical Programming **92**, 257–283.

17. Padberg, M. W., M. R. Rao. 1982. Odd minimum cut-sets and b-matchings. Mathematics of Operations Research **7**, 67–80.
18. Padberg, M., G. Rinaldi. 1991. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Review **33**, 60–100.
19. Tamaki, H. 2003. Alternating cycle contribution: a tour-merging strategy for the travelling salesman problem. Max-Planck Institute Research Report MPI-I-2003-1-007. Saarbrücken, Germany.

Not Every GTSP Facet Induces an STSP Facet

Marcus Oswald, Gerhard Reinelt, and Dirk Oliver Theis

Institute of Computer Science, University of Heidelberg,
Im Neuenheimer Feld 368, 69120 Heidelberg, Germany
{Marcus.Oswald, Gerhard.Reinelt, Dirk.Theis}@informatik.uni-heidelberg.de

Abstract. The graphical traveling salesman problem (GTSP) has been studied as a variant of the classical symmetric traveling salesman problem (STSP) suited particularly for sparse graphs. In addition, it can be viewed as a relaxation of the STSP and employed for solving the latter to optimality as originally proposed by Naddef and Rinaldi. There is a close natural connection between the two associated polyhedra. Until now, it was not known whether there are facets in TT-form of the GTSP polyhedron which are not facets of the STSP polytope as well. In this paper we give an affirmative answer to this question for ≥ 9 and provide a general method for constructing such facets.

Keywords: Graphical Traveling Salesman Problem, Symmetric Traveling Salesman Problem, facets, polyhedral combinatorics, polyhedral computation.

1 Introduction

The *Symmetric Traveling Salesman Polytope,* STSP(n), is the convex hull of all incidence vectors of edge sets of Hamiltonian cycles of the complete graph $K_n = (V_n, E_n)$, where $V_n := \{0, \ldots, n-1\}$ and E_n is the set of all two-element subsets of V_n.

The *Graphical Traveling Salesman Polyhedron,* GTSP(n), is equal to the convex hull of all non-negative integral vectors $(x_e)_{e \in E_n}$ satisfying the properties that the graph with node set V_n and edge set $\{e \mid x_e \neq 0\}$ is connected and that the number $x(\delta(u))$ is even for all $u \in V_n$. Here, for a set $U \subset V_n$, we denote by $\delta(U)$ the set of all edges of K_n with precisely one end node in U, and we define $\delta(u) := \delta(\{u\})$. We also write $x(F) := \sum_{e \in F} x_e$ for a set of edges F.

The Symmetric Traveling Salesman Polytope is at the heart of the well-known cutting-plane method to solve the Symmetric Traveling Salesman Problem [13]. The study of the Graphical Traveling Salesman Polyhedron originates from the attempt to take advantage of the situation when the TSP is defined on non-complete graphs [12, 14]. This line of research is gaining momentum again. In their seminal papers, Naddef and Rinaldi proposed viewing the polyhedron GTSP(n) as a relaxation of STSP(n), the *graphical relaxation* [17, 18]. The motivation for the attempt to use a relaxation instead of the polytope itself is quite obvious in the case of STSP(n) and GTSP(n): while STSP(n) is not even full-dimensional, GTSP(n) is of blocking type and has a number of nice (non-trivial)

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 468–482, 2005.

properties discovered by Naddef and Rinaldi. The observation that STSP(n) is a face of GTSP(n) which is defined by forcing the so-called *degree inequalities* $x(\delta(u)) \geq 2$ to equations is fairly obvious. But Naddef and Rinaldi revealed some surprisingly close connections between the two polyhedra, and raised the following question:

> *Is every TT-form inequality which is facet-defining for GTSP(n) also facet-defining for STSP(n)?* (∗)

Note that almost all GTSP(n)-facets are defined by TT-form inequalities. To be precise, there are only $n(n+1)/2$ facets which are not defined by TT-form inequalities, namely the n degree-facets (the facets defined by the degree inequalities), and the $\binom{n}{2}$ *non-negativity facets* defined by $x_e \geq 0$, $e \in E_n$. But it is known that the total number of facets of, for example, GTSP(10), is at least 51,043,900,866 [1, 11].

Naddef and Rinaldi did not conjecture the answer of (∗) to be *yes,* but recently others have [7]. Being theoretical in origin, in the recent years the issue has increasingly gained practical importance. In [6, 7], a method for separating cuts for the Symmetric TSP is introduced. The inequalities which the method produces define facets of GTSP(n), and in [7] it is conjectured that this implies that they also define facets of STSP(n). In [2, 3, 4], Applegate et al. propose a new method for generating cuts for the Symmetric TSP. In [3], they give an argument supporting the assumption that the resulting cuts are not too weak. Referring to [18], they state that 'it could hardly be better: there is no known counterexample to the conjecture (implicit in Naddef and Rinaldi).'

The outline of this paper is as follows. After explaining some necessary terminology in the next section, we describe a way to prove the existence of TT-type facets of GTSP(n) which do not induce facets of STSP(n) in Section 3. In Section 4, we explain this construction on an example for $n = 9$. In the following section, we explain how this implies that there exist non-NR-facets for all $n \geq 9$. Section 6 contains the computational part of our paper. We explain how we were able to verify computationally that the conjecture is true for $n = 6, 7, 8$, argue why we think that the counterexample we describe is (modulo reordering of nodes) the only one for $n = 9$ and give some facts about counterexamples we found for $n = 10$. Section 7 explains an algorithmic consequence of our results and possible directions of future work.

2 Preliminaries

We assume knowledge of the basics of polyhedral theory (as outlined in e.g. [19]) and also need elementary facts for blocking type polyhedra. We recall that for a polyhedron $P \subset \mathbb{R}^m$ of blocking type, the *blocking polyhedron* is denoted by $B(P) := \{a \in \mathbb{R}^m_+ \mid ax \geq 1 \ \forall x \in P\}$.

For $a \in \mathbb{R}^m$, $\alpha \in \mathbb{R}$ we denote the inequality $ax \geq \alpha$ by (a, α). We write χ^F for the *incidence vector* of F of appropriate length, i.e., $\chi^F_e = 1$, if $e \in F$, and $\chi^F_e = 0$, otherwise. We abbreviate $\chi^{\{f\}}$ by χ^f.

Let P be a polyhedron of dimension d in $\mathbb{R}^m$. A face of dimension $d-2$ is called a *ridge.* A ridge is contained in precisely two facets. Two facets of P are said to be *adjacent on* P, if their intersection is a ridge. A face of dimension one is called an *edge* and a face of dimension zero is called a *vertex* of P. Now let $P \subset \mathbb{R}^m_+$ be a polyhedron of blocking type. The mapping $(a,1) \mapsto a$ defines a canonical bijection from the facet-defining inequalities of P which are not equivalent to non-negativity inequalities $(\chi^j, 0)$ to the vertices of the blocking polyhedron $B(P) := \{a \in \mathbb{R}^m_+ \mid \forall x \in P : ax \geq 1\}$ of P. It also maps valid inequalities defining ridges of P which are not contained in non-negativity facets to points on bounded edges of $B(P)$.

Let $\mathscr{E}(n)$ denote the set of vertices of the graphical TSP polyhedron GTSP(n). It is well-known that GTSP(n) is of blocking type (e.g. [16]). We write $\mathscr{E}^0(n)$ for the set of vertices of GTSP(n) which represent Hamiltonian cycles. These are precisely the vertices of STSP(n), and satisfy $|x|_1 := \sum_e x_e = n$. Further we write $\mathscr{E}^1(n)$ for the set of vertices x of GTSP(n) with $|x|_1 = n+1$ (called *almost Hamiltonian cycles* in [18]), and $\mathscr{E}^{\leq 1}(n) := \mathscr{E}^0(n) \cup \mathscr{E}^1(n)$. From now on, we abbreviate $m := \binom{n}{2}$.

Let $a \in \mathbb{R}^{E_n}$. For a node $u \in V_n$ and an edge $e = \{v_1, v_2\}$ which forms a triangle with u (i.e., $u \notin e$), we define the *shortcut* $\sigma_{u,e} := \chi^{\{v_1,v_2\}} - \chi^{\{u,v_1\}} - \chi^{\{u,v_2\}}$ and $t_{u,e}(a) := -a\,\sigma_{u,e} = a_{\{u,v_1\}} + a_{\{u,v_2\}} - a_{\{v_1,v_2\}}$. Following [18], we define the *TT-set* $\Delta_a(u) := \{e \not\ni u \mid t_{u,e}(a) = 0\}$ for all $u \in V_n$. In other words, the TT-sets can be seen as the sets of shortcuts which are orthogonal to a. A vector a is called *metric,* if its coefficients satisfy the triangle inequality, i.e., if $t(a) \geq 0$, and it is said to be *tight triangular* or *in TT-form,* if for every node u the TT-set $\Delta_a(u)$ is non-empty. Using this, Naddef and Rinaldi [18] proved that the facets of GTSP(n) fall into three categories:

1. non-negativity facets induced by $(\chi^e, 0)$, $e \in E_n$,
2. degree-facets induced by $(\chi^{\delta(u)}, 2)$, $u \in V_n$,
3. facets induced by TT-form inequalities.

We call the facets of the third category *TT-type facets.* We call a TT-type facet of GTSP(n) a *Naddef-Rinaldi facet* or *NR-facet*, if its intersection with STSP(n) is a facet of STSP(n). The question $(*)$ can be restated as: *Is every TT-type facet of GTSP(n) an NR-facet?* We call a TT-type facet F which is not an NR-facet, i.e., for which $\dim(\mathrm{STSP}(n) \cap F) < \dim \mathrm{STSP}(n) - 1 = m - n - 1$ a *non-NR*-facet.

Since the TT-sets Δ_a are invariant under scaling of a with a strictly positive scalar, we can speak of the TT-sets Δ_F of a facet. Let F and G be TT-type facets of GTSP(n). For a node u we say that F and G are *TT-disjoint in* u, if $\Delta_F(u) \cap \Delta_G(u) = \emptyset$. We say that F and G are *nowhere TT-disjoint,* if there is no node in which they are TT-disjoint.

For $a \in \mathbb{R}^m$ and $u \in V_n$ we let $\bar{\lambda}_u(a) := \frac{1}{2} \min_{e \not\ni u} t_{u,e}(a)$. The vector a is metric if and only if the vector $\bar{\lambda}(a) \geq 0$ and it is in TT-form if and only if $\bar{\lambda}(a) = 0$. By Lemma 2.7 of [18] the vector $a - \sum_u \bar{\lambda}_u(a) \chi^{\delta(u)}$ is in TT-form. For an inequality (a, α), we call $(a, \alpha) - \sum_u \bar{\lambda}_u(a)\,(\chi^{\delta(u)}, 2)$ its *TT-form representative.* Naddef and Rinaldi [18] proved the following facts.

a. The degree equations $x(\delta(u)) = 2$, $u \in V_n$ form a complete system of equations for STSP(n). (Therefore, if two inequalities differ by a linear combination of degree equations, they are equivalent.)
b. If an inequality defines a facet of STSP(n), then either it is equivalent to a non-negativity inequality or its TT-form representative defines a facet of GTSP(n).
c. Every facet of STSP(n) which is not defined by a non-negativity inequality is contained in precisely $n+1$ facets of GTSP(n), namely the n degree facets and one NR-facet.
d. Let F be a TT-type facet of GTSP(n). For every $e \in E_n$, there exists $x \in \mathscr{E}^0(n)$ with $x \in F$ and $x_e = 1$.

Item (b) answers the reverse direction of the question $(*)$. Item (c) shows that the relation between the polyhedra GTSP(n) and STSP(n) is very close.

We end this section by citing a result of [18].

Proposition 1 (Corollary 2.6 in [18]). *Let F be a TT-type facet of GTSP(n). For every $e \in E_n$, there exists a $x \in \mathscr{E}^0(n) \cap F$ with $x_e = 1$.*

In other words, the intersection of a TT-facet with STSP(n) is never contained in a non-negativity facet.

3 A Construction for the Graphical TSP

In this section we present the main result of this paper. We start with an observation which motivates our considerations.

Corollary 1. *Let (c, γ) define a TT-type facet F of GTSP(n) which is non-NR. Let $k := \dim STSP(n) - \dim(STSP(n) \cap F)$ denote the co-dimension of STSP(n) $\cap$ F in STSP(n). There exist inequalities $(a_1, \alpha_1), \ldots, (a_k, \alpha_k)$ defining facets of GTSP(n) and STSP(n) (i.e., NR-facets), and $\mu_1, \ldots, \mu_k \geq 0$ such that*

$$(c, \gamma) = \sum_{j=1}^{k} \mu_j (a_j, \alpha_j) - \sum_{u=0}^{n-1} \bar{\lambda}_u \cdot (\chi^{\delta(u)}, 2), \tag{1}$$

where we abbreviate $\bar{\lambda} := \bar{\lambda}\big(\sum_{j=1}^{k} \mu_j a_j\big)$. In other words, (c, γ) is the TT-form representative of $\sum_{j=1}^{k} \mu_j (a_j, \alpha_j)$. The vector $\bar{\lambda}$ is non-negative, and there exists u with $\bar{\lambda}_u > 0$. If $k = 2$, then the (a_i, α) are unique up to scaling.

This is an easy corollary of Proposition 1: Since the face $F \cap \text{STSP}(n)$ is not contained in a non-negativity inequality, any inequality defining the face must be a sum of NR-facet-defining inequalities and degree equations. The degree equations occur with negative coefficient because the weighted sum of the a_j is metric.

Now we describe a construction which allows us to produce non-NR-facets from NR-facets. The basic idea is the following. If two NR-facets G and H are

adjacent on STSP(n), i.e., the face STSP(n) ∩ G ∩ H is a ridge of STSP(n), then it can be seen from the TT-sets of G and H whether they are adjacent on GTSP(n). If they are not, then there exists a non-NR-facet adjacent to G (and, by symmetry, one which is adjacent to H, but the two may be equal). Therefore, in a certain sense, if G and H are not adjacent on GTSP(n), then there is a "hole" between them which is filled by at least one non-NR-facet.

Let (a, α) and (b, β) define NR-facets G and H respectively. Suppose that STSP(n) ∩ G ∩ H is a ridge of STSP(n). For $\mu = (\mu_1, \mu_2) \geq 0$, we let

$$(c_\mu, \gamma_\mu) := \mu_1(a, \alpha) + \mu_2(b, \beta) - \sum_{u=0}^{n-1} \lambda_u(\mu)\,(\chi^{\delta(u)}, 2), \tag{2}$$

where $\lambda(\mu) := \bar{\lambda}\big(\mu_1(a, \alpha) + \mu_2(b, \beta)\big)$. Because of the remarks in the previous section, c is tight triangular. Note that, for all u, if $\lambda_u(\mu) \neq 0$ then G and H are TT-disjoint in u (the converse is true for $\mu_1, \mu_2 > 0$).

Theorem 1. *Let (a, α) and (b, β) define the NR-facets G and H respectively of GTSP(n). If G and H are adjacent on STSP(n) and TT-disjoint at at least one node, then there exists a $\mu > 0$ for which (c_μ, γ_μ) defines a non-NR-facet F of GTSP(n). Moreover, μ can be chosen in such a way that F is adjacent to G (or, by symmetry, in such a way that it is adjacent to H).*

The proof is geometric and fills the remainder of this section. First we will show that the inequality (c_μ, γ_μ) is valid for GTSP(n) and defines a face of dimension at least $m - 2$ of GTSP(n). Then we will prove that if G and H are TT-disjoint in at least one node then there exists μ such that (c_μ, γ_μ) defines a facet F of GTSP(n) which is adjacent to G. Clearly, by construction, the inequality does not define a facet of STSP(n), because it is dominated by (a, α) and (b, β). Hence, F is a non-NR-facet.

Lemma 1. *The inequality (c_μ, γ_μ) is valid for GTSP(n).*

Proof. Since c_μ is metric, the validity of (c_μ, γ_μ) for GTSP(n) follows from its validity for STSP(n), which is given because it is a sum of valid inequalities.

Let R_μ denote the face of GTSP(n) defined by (c_μ, γ_μ).

Proposition 2. *R_μ is "at least a ridge", i.e.,* $\dim R_\mu \geq \dim$ *GTSP(n)* $- 2$.

The argument we use is the same as that in the proof of Theorem 2.11 of [18].

Proof. First note that STSP(n)∩G∩H ⊂ R_μ, and choose an affinely independent subset $x^1, \ldots, x^{m-n-2}$ of $\mathscr{E}^0(n) \cap G \cap H$. Now let $u \in V_n$ and $e \in \Delta_{c_\mu}(u)$. Because STSP(n) ∩ G ∩ H is a ridge of STSP(n), it is not contained in the non-negativity facet defined by $(\chi^e, 0)$. Thus there exists an $x \in \mathscr{E}^0(n) \cap G \cap H$ with $x_e = 1$. Then $z^u := x - \sigma_{u,e} \in R_\mu$.

We now have constructed $m - 2$ vectors, and it remains to be shown that they are affinely independent. For this, let $\xi_1, \ldots, \xi_{m-n-2}$ and $\zeta_0, \ldots, \zeta_{n-1}$ be given in such a way that $\sum_j \xi_j + \sum_u \zeta_u = 0$ and

$$\sum_{j=1}^{m-n-2} \xi_j x^j + \sum_{u=0}^{n-1} \zeta_u z^u = 0.$$

Let $v \in V_n$. By applying $\delta(v)$ to both sides of the equation we obtain

$$0 = 2 \sum_{j=1}^{m-n-2} \xi_j + 2 \sum_{u=0}^{n-1} \zeta_u + 2\zeta_v,$$

which implies $\zeta_v = 0$. Since the x^j are affinely independent, it follows from $\zeta_v = 0$ for all v that $\xi_j = 0$ for all j. This completes the proof of the proposition.

As a consequence of this proposition we obtain the following corollary.

Corollary 2. *If G and H are NR-facets which are adjacent on STSP(n), then they are adjacent on GTSP(n) if and only if they are nowhere TT-disjoint.*

Proof. It is easy to see that if G and H are TT-disjoint at a node u then the degree facet for u contains $G \cap H$, whence G and H cannot be adjacent. If, on the other hand, G and H are TT-disjoint, then we have $\lambda = 0$ in (2). Therefore, by the proposition, with $\mu := (\frac{1}{2}, \frac{1}{2})$, $G \cap H = R_\mu$, is a ridge.

To this point we have not made any restrictions concerning the TT-sets of G and H. But now we will require that G and H must be TT-disjoint in at least one node, i.e., $\forall \mu : \lambda(\mu) \neq 0$.

Lemma 2. *γ_μ never vanishes, i.e., for all $\mu \geq 0$ we have $\gamma_\mu > 0$.*

Proof. From [17] (or from the fact that GTSP(n) is of blocking type) it is known that the only facets of GTSP(n) which are defined by inequalities whose right hand side vanishes, are the non-negativity facets. Hence, if $\gamma_\mu = 0$ for a μ, then R_μ is the intersection of the non-negativity facets on the edges e on which c_μ is non-zero (this may be none at all). Consequently, STSP(n) $\cap\, G \cap H$ is the intersection of degree facets and non-negativity facets, which is a contradiction.

We now come to the heart of our geometric existence argument. With the abbreviation $\mu(s) := (1-s,\ s)$ and because of the previous lemma, we can define the following mapping:

$$\phi\colon [0,1] \to \mathbb{R}^{E_n} : s \mapsto \frac{1}{\gamma_{\mu(s)}} \cdot c_{\mu(s)}.$$

Lemma 3. *ϕ is continuous.*

Proof. The mappings $\mu \mapsto t_{u,e}(\mu_1 a + \mu_2 b)$ for $e \not\ni u$ are continuous. Thus $\mu \mapsto \lambda(\mu)$ is continuous as a minimum of continuous mappings. Invoking Lemma 2, ϕ is also continuous.

Now we are ready to complete the proof of the main result of this paper.

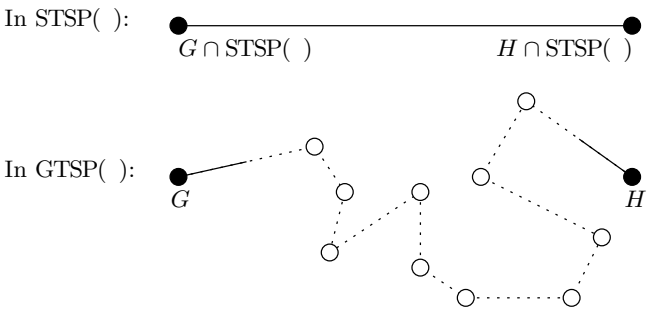


Fig. 1. Curve on the boundary of $B(\text{GTSP}(\ ))$ defined by adjacent facets of STSP( )

Proof (of Theorem 1). The mapping ϕ defines a curve on the boundary of the blocking polyhedron $B(\text{GTSP}(n))$. The fact that $(\phi(s), 1)$ defines a face of dimension at least $m-2$ of GTSP(n) which is not contained in a non-negativity face is equivalent to the fact that the image $\phi(s)$ is always contained in a bounded edge of $B(\text{GTSP}(n))$. The curve starts in the vertex of $B(\text{GTSP}(n))$ corresponding to G and ends in the vertex corresponding to H. Because G and H are not adjacent on GTSP(n), these vertices are not adjacent in the graph of $B(\text{GTSP}(n))$. Since ϕ is continuous, it must traverse a third vertex of $B(\text{GTSP}(n))$. This proves that there exists an s for which $(\phi(s), 1)$ defines a facet of GTSP(n). The facet defined by the smallest such s is adjacent to G. This completes the proof of Theorem 1.

For an illustration, see Fig. 1. The facets $H \cap \text{STSP}(n)$ and $G \cap \text{STSP}(n)$ are adjacent on STSP(n). In the case of GTSP(n), ϕ is a curve on the boundary of $B(\text{GTSP}(n))$, which connects G to H and walks along edges of the blocking polyhedron. The points represent vertices of $B(\text{GTSP}(n))$. In the hollow points, $(\phi(s), 1)$ defines a non-NR-facet of GTSP(n).

4 An Example

We have not yet shown that there exists a non-NR-facet, but now we know how to search for one. In this section we will explain the construction on an example. This example satisfies a certain condition, which makes the geometry particularly nice. The continuous curve ϕ on the boundary of the blocking polyhedron can be visualized as continuously "tilting" a valid halfspace. We will see that the condition (3) below implies the existence of ϱ for which the intersection of the halfspace with GTSP(n) is contained in G if $\mu_1/\mu_2 < \varrho$ and in H if $\mu_1/\mu_2 > \varrho$. For $\mu_1/\mu_2 = \varrho$, the halfspace defines a face of GTSP(n) which contains two ridges: a facet.

Let $(\dot{a}, 18)$ and $(\dot{b}, 28)$ be the two inequalities whose coefficients are displayed in Table 1. They are taken from the list of facets of STSP(n), $n = 6, \ldots, 10$ (see [10]). Since they are in TT-form, they define NR-facets by item (b) in Section 2. By enumerating all $x \in \mathscr{E}^0(9)$ it is easy to see that they are adjacent

Table 1. Two inequalities defining Naddef-Rinaldi-facets

	1	2	3	4	5	6	7	8	
0	4	3	3	2	3	5	1	2	
1		1	3	2	5	3	3	2	
2			2	3	4	4	2	3	
3				1	4	4	2	3	
4					3	5	3	4	
5						2	4	5	
6							4	3	
7								1	≥ 18

	1	2	3	4	5	6	7	8	
0	6	4	4	6	4	6	2	3	
1		2	6	4	6	4	4	3	
2			4	6	8	6	6	5	
3				2	8	6	6	5	
4					6	8	6	7	
5						2	4	5	
6							4	3	
7								1	≥ 28

on STSP(9). They are TT-disjoint in node 0 but nowhere else. The results of the previous section now imply the existence of a non-NR-facet for GTSP(9).

We want to construct this counterexample explicitly. By inserting the definition of λ and $\bar{\lambda}$ in (2), and using the fact that $t(\cdot)$ is linear, we have

$$(c_\mu, \gamma_\mu) = \mu_1(\dot{a}, 18) + \mu_2(\dot{b}, 28) - \tfrac{1}{2} \min_{e \not\ni 0} \big(\mu_1 t_{0,e}(\dot{a}) + \mu_2 t_{0,e}(\dot{b})\big) \cdot (\chi^{\delta(0)}, 2).$$

Looking at the integers $t_{0,e}(\dot{a})$ and $t_{0,e}(\dot{b})$ for $e \not\ni 0$, we find that they are all either zero or ≥ 2 and that $\min\{t_{0,e}(\dot{a}) \mid e \not\ni 0,\, t_{0,e}(\dot{b}) = 0\} = 2$ (and the same holds when $\dot{a}$ and $\dot{b}$ are exchanged in the formula). But this immediately implies the following relation for $a := \dot{a}$, $b := \dot{b}$ and $u := 0$

$$m_b t_{u,e}(a) + m_a t_{u,e}(b) \geq m_a m_b \quad \text{for all } e \not\ni u \tag{3}$$

where, for arbitrary a and b the number m_a is defined as

$$m_a := \min\{t_{u,e}(a) \mid e \not\ni u,\, t_{u,e}(b) = 0\},$$

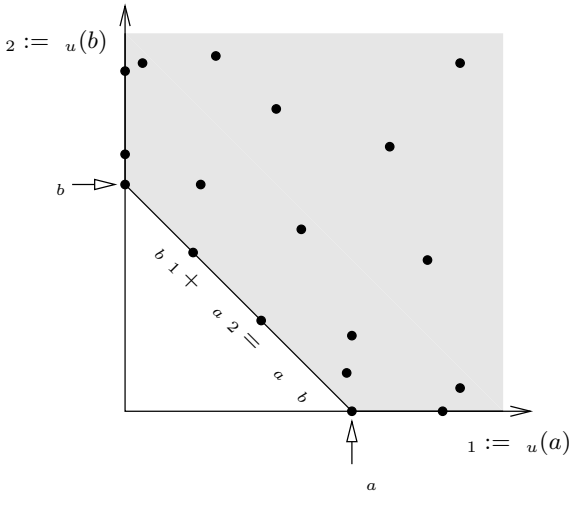


Fig. 2. Graphical illustration of relation (3)

and m_b symmetrically. See Fig. 2 for an illustration of this relation. In this figure, the black dots represent the points $(t_1, t_2) := (t_{u,e}(a), t_{u,e}(b))$, $e \not\ni u$. For this situation we can give the following general result.

Proposition 3. *Let (a, α) and (b, β) be as in Theorem 1, but TT-disjoint in precisely one node u. Define $\varrho := m_b/m_a$. If (3) holds, then (c_μ, γ_μ) defines a facet of GTSP(n) if and only if $\mu_1/\mu_2 = \varrho$. This facet is adjacent on GTSP(n) to both G and H.*

Proof. The proof is in three steps. *1.)* First we show that, for all $\mu \geq 0$ with $\mu_1/\mu_2 > \varrho$, if the minimum value over e of $\mu_1 t_{u,e}(a) + \mu_2 t_{u,e}(b)$ is attained in f then $t_{u,f}(a) = 0$. To see this, consider the set of points $S := \{(t_{u,e}(a), t_{u,e}(b)) \mid e \not\ni u\}$ in $\mathbb{R}^2_+$. The relation (3) means that no point is below the line defined by $m_b t_1 + m_a t_2 = m_a m_b$ (cf. Fig. 2). But, by definition of m_a and m_b, the points $(m_a, 0)$ and $(0, m_b)$ are in S and on the line. Minimizing a linear function (μ_1, μ_2) over the set S will give either the unique point $(m_a, 0)$, if $\mu_1/\mu_2 < \varrho$, or the unique point $(0, m_b)$, if $\mu_1/\mu_2 > \varrho$, or all points of S on the line, if $\mu_1/\mu_2 = \varrho$.

2.) Second, we show that if $\mu_1/\mu_2 > \varrho$, then $R_\mu \subset G$ holds. Namely, we prove that $x \in \mathscr{E}(n) \cap R_\mu$ implies $x \in G$, and we do this by induction on $|x|_1$. If $|x|_1 = n$, i.e., if x is a Hamiltonian cycle, the claim is true, because $R_\mu \cap \mathrm{STSP}(n) = G \cap H \cap \mathrm{STSP}(n)$. For the induction step, let $x \in R_\mu$ be a vertex of GTSP(n) with $|x|_1 \geq n + 1$. From Lemma 2.5 of [18], it is known that there exists a $y \in R_\mu$ with $|y|_1 = |x|_1 - 1$ and such that $y - x$ is a shortcut $\sigma_{v,f}$. We can conclude that $f \in \Delta_v(c_\mu)$. If we can show that $f \in \Delta_v(a)$, then $x \in G$ implies $y \in G$. In the case $v \neq u$ we clearly have $f \in \Delta_v(a) \cap \Delta_v(b)$. In the case $v = u$, the minimum value over e of $\mu_1 t_{u,e}(a) + \mu_2 t_{u,e}(b)$ is attained in f, and thus, by 1., we have $t_{u,f}(a) = 0$, whence the claim is established.

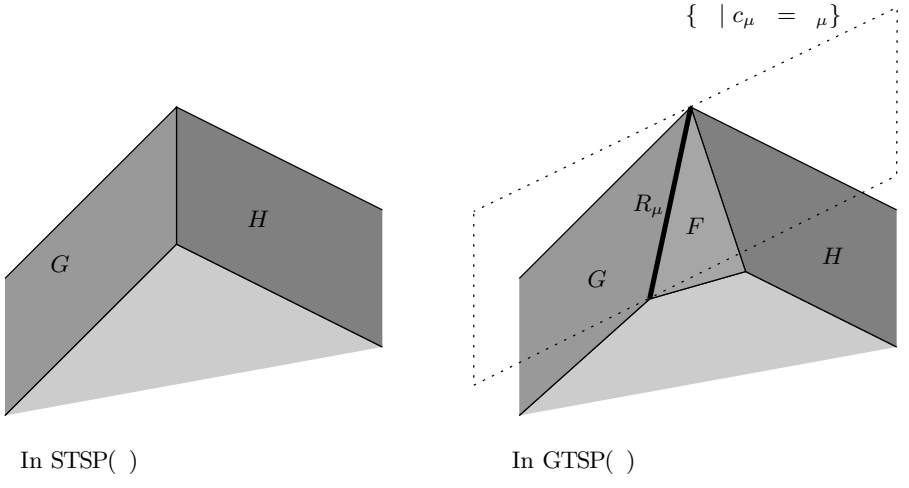
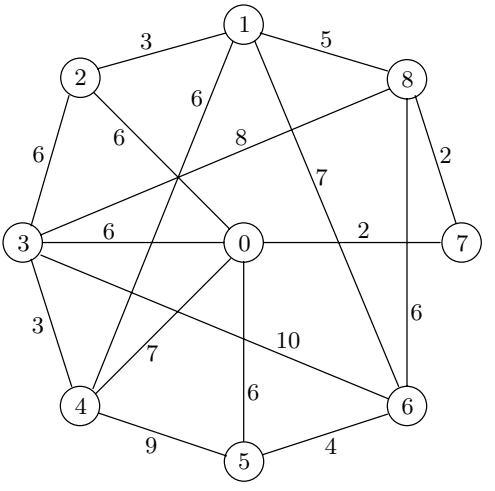


Fig. 3. Tilting a hyperplane



	1	2	3	4	5	6	7	8
0	9	6	6	7	6	10	2	4
1		3	9	6	11	7	7	5
2			6	9	12	10	8	8
3				3	12	10	8	8
4					9	13	9	11
5						4	8	10
6							8	6
7								2

Fig. 4. Counterexample to the conjecture $(*)$ that every TT-type facet of GTSP( ) induces a facet of STSP( ) for $= 9$

3.) Now we can conclude that (c_μ, γ_μ) defines a hyperplane which is tilted continuously. For $\mu_1/\mu_2 > \varrho$ its intersection with GTSP(n) is a ridge contained in G, for $\mu_1/\mu_2 < \varrho$ the intersection is a ridge contained in H. Since it is moved continuously, the intersection contains both ridges for $\mu_1/\mu_2 = \varrho$, which implies that it must be a facet. See Fig. 3 for an illustration. In this figure, the image on the left shows the situation on STSP(n): the two facets $G \cap$ STSP(n) and $H \cap$STSP(n) are adjacent. The image on the right shows how the valid hyperplane (dotted rectangle) intersects GTSP(n) in a ridge R_μ linking G and the non-NR-facet F. If the hyperplane is tilted, the intersecion with GTSP(n) will be equal to the non-NR-facet F before it is the ridge linking F and H. With this, the proof of the proposition is completed.

As a consequence of this proposition, with $\varrho = 1$ and $\mu_1 = \mu_2 = 1$ the inequality $(\dot{a}, 18) + (\dot{b}, 28) - (\chi^{\delta(u)}, 2)$ is the non-NR-facet for $n = 9$ we were looking for. It is displayed in Fig. 4. In this figure, the remaining edges have coefficients which are equal to the lengths of the shortest paths. The right hand side is 44.

5 Existence of Non-NR-Facets for $n \geq 9$

In the previous section we have deployed a non-NR-facet for $n = 9$ based on the technique introduced in Section 3. We will now conclude the theoretical part of this paper by proving that there exist non-NR-facets for all $n \geq 9$. For this we use 0-node lifting, as defined in [17, 18].

Let $a \in \mathbb{R}^{V_n}$ and $w \in V_n$. We define $a^\circ \in \mathbb{R}^{V_{n+1}}$ as follows:

$$a^\circ_e = \begin{cases} a_e, & \text{if } n \notin e \\ a_{\{u,w\}}, & \text{if } e = \{u,n\} \text{ and } u \neq w \\ 0 & \text{if } e = \{w,n\}. \end{cases}$$

In [17, 18], it is shown that if (a,α) defines a TT-type facet of GTSP(n), then (a°,α) defines a TT-type facet of GTSP($n+1$). We call (a°,α) the inequality obtained by *0-node-lifting at node u.*

Theorem 2. *For all $n \geq 9$ there exists a non-NR-facet of GTSP(n).*

Proof. We have just displayed the case $n = 9$. For $n \geq 10$, we use induction based on 0-node lifting. Let $(c,\gamma) = \mu_1(a,\alpha) + \mu_2(b,\beta) - \sum_{u=0}^{n-1} \lambda_u(\chi^{\delta(u)}, 2)$, and $w \in V_n$ with $\lambda_w = 0$. Denote by (c°,γ) the inequality obtained from (c,γ) by 0-node lifting at w.

Claim: If (c,γ) defines a non-NR-facet, then so does (c°,γ).

From the claim the statement of the theorem follows, because the non-NR-facet described in the previous section is of this form.

Proof of the claim. As mentioned above, (c°,γ) defines a TT-type facet of GTSP($n+1$). It is not a NR-facet, because on STSP($n+1$) it is dominated by the inequalities (a°,α) and (b°,β), i.e., by the inequalities obtained from (a,α) and (b,β) respectively by 0-node lifting at node w. This completes the proof of the theorem.

6 Computational Findings and the Case $n \leq 8$

In this section we describe how we verified computationally that for $n \leq 8$ there is no non-NR-facet of GTSP(n), and how we "confirmed" the theoretical results of Sections 3 to 5 for $n = 9, 10$.

It follows from [15] that the answer to $(*)$ is *yes* for $n \leq 5$. We applied the facet enumeration software PORTA [9] in a standard way to check the correctness of the conjecture for $n = 6$ by computing all facets of GTSP(6). However, the computation did not terminate within 50 days and we had to give up. The numbers of vertices of the GTSP(n) polyhedra for $n = 3, \ldots, 10$, as shown in Table 2, indicate that any naive attempt in this direction is likely to fail (the middle column in the table refers to Lemma 4).

We do not know of any result on complete descriptions for GTSP(n) for $n \geq 6$, either theoretically or computationally. The complete description of STSP(n) for $n = 6, 7$ was proven in [5], for $n = 8$ it was computed in [8], and for $n = 9$ it was computed in [11]. For $n = 10$ the same system of inequalities was produced in [1, 11], and it is conjectured to be complete in [11].

We used the following method to verify computationally that every TT-type facet of GTSP(n) is an NR-facet for $n = 6, 7, 8$. As mentioned in [18], every TT-type facet contains an $x \in \mathscr{E}^0(n)$. By reordering the nodes, we may fix one

Table 2. Number of vertices of STSP( ) and GTSP( )

	STSP		GTSP
	$\lvert\mathscr{E}^0(\ )\rvert$	$\lvert\mathscr{E}^{\leq 1}(\ )\rvert$	$\lvert\mathscr{E}(\ )\rvert$
3	1	4	4
4	3	15	31
5	12	87	362
6	60	600	5676
7	360	4770	111982
8	2520	42840	$\geq$2669752
9	20160	428400	$\geq$74896444
10	181440	4717440	$\geq$2388208240

$x_0 \in \mathscr{E}^0(n)$, and consider the cone C defined by the rays $x - x_0$ where $x \in \mathscr{E}(n)$. Every TT-facet is an NR-facet if and only if this cone is equal to the cone C' described by the TT-form inequalities defining facets of STSP(n) for which x_0 is satisfied with equality, plus appropriate non-negativity and degree inequalities. The system of inequalities for C' was computationally tractable for $n = 6, 7, 8$: the computer program PORTA produced a complete list of extreme rays of C'. After appropriately scaling the rays and adding them to x_0, we obtained points in GTSP(n). Hence, $C' \subset C$ holds. From item (b) in Section 2 it follows that every facet of C' is also a facet of C. This implies $C = C'$. Therefore, as a result of our computations, we gain the following proposition.

Proposition 4. *For $n \leq 8$ every TT-type facet of GTSP(n) is an NR-facet.*

Since for $n = 9$ the cone C' has about 88000 facets, we tried a refinement of this idea. It is easy to see that every TT-facet must contain an element of $\mathscr{E}^1(n)$. There are precisely four non-isomorphic members of $\mathscr{E}^1(n)$. The four cones defined analogously to C' above had between 500 and 8000 facets. Only the cone with the smallest number of facets was tractable, but all four would have to be screened. Therefore we had to give up on this method for $n = 9$.

How else can we look for non-NR-facets computationally? Of course—one possibility is to compute TT-type facets of GTSP(n) and check the dimension of the intersection with STSP(n). The complete outer description of STSP(n) is known for $n \leq 9$ and conjectured for $n = 10$ [11]. With it we know all NR-facets of GTSP(n), $n \leq 9$ and many of GTSP(10). This suggests that we identify the facets adjacent to these NR-facets on GTSP(n), $n \leq 10$ using the adjacency decomposition technique described in [11]. This technique uses the fact that the facets of GTSP(n) which are adjacent to G can be derived from the facets of G (which we computed using PORTA) by sequential lifting. To reduce the computational effort, the following lemma turned out to be helpful, the proof of which we omit.

Lemma 4. *Let $P := \operatorname{conv} \mathscr{E}^{\leq 1}(n) + \mathbb{R}^m_+$. If G and H are two adjacent facets of GTSP(n), then $G \cap P$ and $H \cap P$ are adjacent facets of P.*

A consequence of the lemma is that, computing the outer description of G, it is sufficient to use the polyhedron P, which has many fewer vertices than GTSP(n) on G. See Table 2, where the middle column gives the number of vertices of the polyhedron P.

For $n = 9$, using the adjacency decomposition described above, we have found precisely one class of non-NR-facets, namely the one described in Section 4. The class of a facet consists of all facets arising from one by renumbering the nodes arbitrarily. We managed to compute the neighbors of 189 out of 192 classes of NR-facets of GTSP(9). Only the three facet-classes induced by subtour inequalities $(\delta(U), 2)$ where $U \subsetneq V_9$, $2 \leq |U| \leq 4$, remain. Since the results of Sections 3 and 4 relate the inequalities (a_i, α) of (1) to adjacency on GTSP(n), and it is possible to show that no non-NR-facet can be written in the form (1) by subtours only, we conjecture that we have produced the only non-NR-facet class (and thus a complete description of GTSP(9)).

For $n = 10$ we found 49 classes of non-NR-facets. In almost all cases the intersection of the non-NR-facets with STSP(10) defines a ridge of STSP(10), conforming to the construction of Section 3.

We have also found three classes of non-NR-facets (all with $n = 10$) where the dimension of the intersection with STSP(10) falls below the dimension of a ridge. More precisely, in each of the three cases the dimension is 32 while $\dim \mathrm{STSP}(10) = 35$ and the dimension of a ridge is 33. But we assume that there are further non-NR-facets where this gap to the dimension of STSP(n) is even bigger than three.

7 Conclusion and Outlook

We note that the result of this paper shows that the degree constraints can affect the value of an LP-relaxation even if the cost function satisfies the triangle inequality. This answers a question posed by Goemans [16].

As mentioned in the introduction, the so-called *local-cuts* method of [3, 4] produces valid inequalities for Symmetric Traveling Salesman Polytopes. They define facets of GTSP(n) for a small n (up to about 30), and are then "lifted". In [3], the formulation of the conjecture is stronger than the formulation $(*)$. This is mainly so because only a subset of vertices of GTSP(n) are considered, namely the set of vectors which are called *strongly constrained tangled tours* in [3, 4]. The set of strongly constrained tangled tours is defined by a fractional vector. It can be seen that our non-NR-facet for $n = 9$ (Fig. 4) defines a facet of the convex hull of all strongly constrained tangled tours which are defined by the fractional vector displayed in Fig. 5. This implies that the local-cuts method may produce inequalities which are not facet-defining for STSP(n) even for $n = 9$. Given that the number of non-NR-facets increases with n, it would be very interesting to know if the local-cuts method produces many non-facet-defining inequalities for the Symmetric Traveling Salesman Polytope.

It would also be interesting to know how big the co-dimension $\dim \mathrm{STSP}(n) - \dim(\mathrm{STSP}(n) \cap F)$ can be for a facet F of GTSP(n). While we have no theoretical

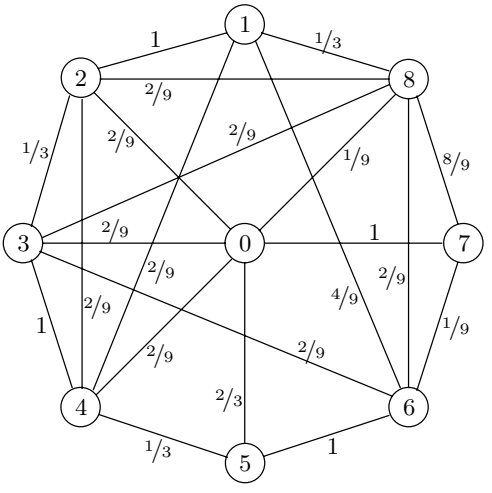


Fig. 5. Fractional vector satisfying all STSP(9)-facet-defining inequalities and the degree equations for all nodes except 0

insights which would support such a conjecture, looking at the examples we have, one would be tempted to guess that

$$g_n := \max\{\dim \mathrm{STSP}(n) - \dim(\mathrm{STSP}(n) \cap F) \mid F \text{ facet of } \mathrm{GTSP}(n)\} \leq n + 1.$$

The next step in the investigation of the graphical relaxation will be to check another open question raised in [18], which is concerned with the 0-node-lifting operation. As noted above, Naddef and Rinaldi [18] showed that 0-node-lifting preserves the facet-defining property for GTSP. The question which remains open to this date is: *Does 0-node-lifting preserve NR-facets?* If the answer to $(*)$ had been *yes,* then the 0-node-lifting question would have had a positive answer, too. Hence, after the findings of this paper, this has become less likely. The 0-node-lifting question is of high relevance both in theory *and* in practice. To answer it, a characterization of NR-facets will be required.

References

1. D. Alevras and M. W. Padberg. *Linear optimization and extensions: problems and solutions.* Springer-Verlag Berlin Heidelberg, 2001.
2. D. Applegate, R. Bixby, V. Chvátal, and W. Cook. On the solution of the Traveling Salesman Problem. In *Doc. Math. J. DMV (Extra Volume ICM)*, pages 645–656, 1998.
3. D. Applegate, R. Bixby, V. Chvátal, and W. Cook. TSP cuts which do not conform to the template paradigm. In M. Jünger and D. Naddef, editors, *Computational Combinatorial Optimization*, pages 261–303. Springer-Verlag Berlin Heidelberg, 2001.

4. D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Implementing the Dantzig-Fulkerson-Johnson algorithm for large Traveling Salesman Problems. *Math. Program. Ser. B*, 97(1–2):91–153, 2003.
5. S. C. Boyd and W. H. Cunningham. Small Travelling Salesman Polytopes. *Math. Oper. Res*, 16(2):259–271, 1991.
6. R. Carr. Separating over classes of TSP inequalities defined by 0-node lifting in polynomial time. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Proc. IPCO V*, pages 460–474, 1996.
7. R. Carr. Separation algorithms for classes of STSP inequalities arising from a new STSP relaxation. *Math. Oper. Res*, 29(1):80–91, 2004.
8. T. Christof, M. Jünger, and G. Reinelt. A complete description of the traveling salesman polytope on 8 nodes. *OR Letters*, 10:497–500, 1991.
9. T. Christof and A. Löbel. PORTA – a polyhedron representation algorithm, 1998. `http://www.informatik.uni-heidelberg.de/groups/comopt/software/PORTA/`.
10. T. Christof and G. Reinelt. SmaPo – library of Small Polytopes. `http://www./informatik.uni-heidelberg.degroups/comopt/software/SMAPO//tsp/tsp.html`.
11. T. Christof and G. Reinelt. Decomposition and parallelization techniques for enumerating the facets of combinatorial polytopes. *Int. J. Comput. Geom. Appl.*, 11: 423–437, 2001.
12. G. Cornuéjols, J. Fonlupt, and D. Naddef. The Traveling Salesman Problem on a Graph and some related Integer Polyhedra. *Math. Program.*, 33:1–27, 1985.
13. G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling salesman problem. *Oper. Res.*, 2:393–410, 1954.
14. B. Fleischmann. A cutting plane procedure for the travelling salesman problem on road networks. *Eur. J. Oper. Res.*, 21:307–317, 1985.
15. J. Fonlupt and D. Naddef. The Traveling Salesman Problem in graphs whith some excluded minors. *Math. Program.*, 53(2):147–172, 1992.
16. M. X. Goemans. Worst-case comparison of valid inequalities for the TSP. *Math. Program.*, 69(2):335–349, 1995.
17. D. Naddef and G. Rinaldi. The symmetric traveling salesman polytope and its graphical relaxation: Composition of valid inequalities. *Math. Program.*, 51:359–400, 1991.
18. D. Naddef and G. Rinaldi. The graphical relaxation: A new framework for the Symmetric Traveling Salesman Polytope. *Math. Program.*, 58:53–88, 1993.
19. A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986.

Author Index