Oktay Günlük
Gerhard J. Woeginger (Eds.)

LNCS 6655

Integer Programming and Combinatorial Optimization

15th International Conference, IPCO 2011
New York, NY, USA, June 2011
Proceedings

IPCO 2011

Springer

Lecture Notes in Computer Science 6655

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

Oktay Günlük Gerhard J. Woeginger (Eds.)

Integer Programming and Combinatoral Optimization

15th International Conference, IPCO 2011
New York, NY, USA, June 15-17, 2011
Proceedings

Springer

Volume Editors

Oktay Günlük
IBM T. J. Watson Research Center
Mathematical Sciences Department
Yorktown Heights, NY 10598, USA
E-mail: gunluk@us.ibm.com

Gerhard J. Woeginger
TU Eindhoven
Department of Mathematics and Computer Science
P.O. Box 513
5600 MB, Eindhoven, The Netherlands
E-mail: gwoegi@win.tue.nl

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-20806-5 e-ISBN 978-3-642-20807-2
DOI 10.1007/978-3-642-20807-2
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011925954

CR Subject Classification (1998): F.2, E.1, I.3.5, G.2, G.1.6, F.2.2

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

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

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

Preface

This volume contains the 33 papers presented at IPCO 2011, the 15th Conference on Integer Programming and Combinatorial Optimization, held during June 15–17, 2011 at the IBM T.J. Watson Research Center in New York, USA. IPCO conferences are sponsored by the Mathematical Optimization Society.

The first IPCO conference took place at the University of Waterloo in May 1990. A lot of things have changed over the course of the 20 years since then. IPCO has become extremely visible, and it has grown into one of the main events for the mathematical programming community. IPCO has also become bigger in size (for instance, the number of submissions has increased from 76 to 110, and the number of Program Committee members has increased from 8 to 14). And IPCO has become considerably broader in scope (nowadays the meeting covers topics that did not even exist in 1990). Of course there are other things that remained constant over time. For instance, the format of IPCO is still a single stream of non-parallel sessions (this has become IPCO tradition, and is unlikely to change in the near future). The number of accepted papers still is 33 (plus/minus 3). And – most important – the quality of the presented research is at the very very top.

Altogether IPCO 2011 received 110 extended abstracts. Each submission was reviewed by at least three, and on average by four, Program Committee members prior to the committee meeting in January 2011 in Aussois. During the committee meetings all 110 submission were discussed thoroughly and 33 papers were selected for presentation. We expect the full versions of the papers contained in this volume to be submitted for publication in refereed journals.

Many people contributed to the smooth running and the success of IPCO 2011. In particular our thanks go to:

- All authors who submitted their current research to IPCO
- Our reviewers and subreferees whose expertise flowed into the decision process
- The members of the Program Committee who graciously gave their time and energy
- The members of the Local Organizing Committee who made the conference possible
- The organizers of the 2011 Aussois workshop on combinatorial optimization (Michael Jünger, Gerhard Reinelt, Giovanni Rinaldi) for kindly giving us the opportunity to have a physical Program Committee meeting
- The EasyChair conference management system for hosting the evaluation process

June 2011

Oktay Günlük
Gerhard J. Woeginger

Organization

Program Committee

Nikhil Bansal	IBM T.J. Watson Research Center, USA
Michele Conforti	Universitá degli Studi di Padova, Italy
Bertrand Guenin	University of Waterloo, Canada
Oktay Günlük	IBM T.J. Watson Research Center, USA
Tibor Jordán	Eötvös University, Budapest, Hungary
Jochen Könemann	University of Waterloo, Canada
Andrea Lodi	DEIS, University of Bologna, Italy
Franz Rendl	University of Klagenfurt, Austria
Giovanni Rinaldi	IASI – CNR, Italy
Günter Rote	Freie Universität Berlin, Germany
Cliff Stein	Columbia University, USA
Frank Vallentin	TU Delft, The Netherlands
Jens Vygen	University of Bonn, Germany
Gerhard Woeginger	TU Eindhoven, The Netherlands

Organizing Committee

Sanjeeb Dash
Oktay Günlük
Jon Lee
Maxim Sviridenko

External Reviewers

Aardal, Karen	Achterberg, Tobias
Ahmed, Shabbir	Archer, Aaron
Atamturk, Alper	Bachoc, Christine
Barvinok, Alexander	Bateni, Mohammad Hossein
Beck, Matthias	Bhatnagar, Nayantara
Bienstock, Dan	Bogart, Tristram
Bonami, Pierre	Borodin, Allan
Bosse, Hartwig	Brandstädt, Andreas
Buchbinder, Niv	Byrka, Jaroslaw
Büsing, Christina	Caprara, Alberto
Chakrabarty, Deeparnab	Chan, Yuk Hei
Cheriyan, Joseph	Cornuéjols, Gérard
D'Ambrosio, Claudia	D'Andreagiovanni, Fabio

Dash, Sanjeeb
Dutour Sikirić, Mathieu
Dür, Mirjam
Faenza, Yuri
Felsner, Stefan
Fleiner, Tamás
Frank, András
Gärtner, Bernd
Garg, Naveen
Geleji, János
Georgiou, Konstantinos
Gouveia, Luis
Grant, Elyot
Gupta, Shubham
Hajiaghayi, Mohammad Taghi
Held, Stephan
Henrion, Didier
Hurkens, Cor
Ishii, Toshimasa
Iyengar, Garud
Jüttner, Alpár
Katoh, Naoki
Kim, Edward D.
Kis, Tamás
Korula, Nitish
Krishnaswamy, Ravishankar
Laurent, Monique
Levin, Asaf
Li, Yanjun
Liberti, Leo
Louveaux, Quentin
Mahjoub, Ridha
Makino, Kazuhisa
Manlove, David F
Maßberg, Jens
Miklós, Zoltán
Mitchell, John
Murota, Kazuo
Nguyen Kim, Thang
Nutov, Zeev
Olver, Neil
Ouorou, Adam
Pap, Júlia
Peis, Britta
Pfetsch, Marc
Di Summa, Marco
Dyer, Martin
Epstein, Leah
Fekete, Sandor
Fleiner, Balázs
Frangioni, Antonio
Fukasawa, Ricardo
Galli, Laura
Geelen, Jim
Gentile, Claudio
Gerards, Bert
Grandoni, Fabrizio
Grappe, Roland
Gvozdenović, Nebojša
Harvey, Nick
Helmberg, Christoph
Hildebrand, Martin
Hähnle, Nicolai
Iwata, Satoru
Jansen, Klaus
Kaibel, Volker
Kellerer, Hans
Király, Csaba
Kiwiel, Krzysztof
Král, Daniel
Labbé, Martine
Lejeune, Miguel
Li, Fei
Li, Zhentao
Linderoth, Jeff
Lozin, Vadim
Makarychev, Konstantin
Malaguti, Enrico
Marx, Dániel
Mathieu, Claire
Mirrokni, Vahab
Monaci, Michele
Nagarajan, Viswanath
Niemeier, Martin
Oliveira, Fernando
Oriolo, Gianpaolo
Pap, Gyula
Parekh, Ojas
Pendavingh, Rudi
Phillips, David

Pivotto, Irene
Pritchard, David
Ralph, Daniel
Roberti, Roberto
Römisch, Werner
Sanità, Laura
Schürmann, Achill
Sebő, András
Sethuraman, Jay
Singh, Mohit
Smith, Cole
Srinivasan, Aravind
Sviridenko, Maxim
Swamy, Chaitanya
Szigeti, Zoltán
Thomas, Rekha
Tunçel, Levent
Van Vyve, Mathieu
Verschae, José
Vielma, Juan Pablo
Vondrak, Jan
Végh, László
Wolkowicz, Henry
Wolsey, Laurence
Yaman, Hande
Zenklusen, Rico
Zhang, Lisa
Post, Ian
Raack, Christian
Ravi, R.
Rothvoß, Thomas
Salvagnin, Domenico
Scheideler, Christian
Schultz, Rüdiger
Sen, Suvrajeet
Shmoys, David
Skutella, Martin
Spieksma, Frits
Svensson, Ola
Svitkina, Zoya
Szabó, Jácint
Theis, Dirk Oliver
Tramontani, Andrea
Van Der Vlerk, Maarten
Ventura, Paolo
Vetta, Adrian
Von Heymann, Frederik
Vredeveld, Tjark
Wiegele, Angelika
Wollan, Paul
Xiao, Lin
Zambelli, Giacomo
Zhang, Guochuan
Ziegler, Günter M.

Table of Contents

An Excluded Minor Characterization of Seymour Graphs

Alexander Ageev[1], Yohann Benchetrit[2], András Sebő[2], and Zoltán Szigeti[2]

[1] Sobolev Institute of Mathematics, Novosibirsk, Russia
[2] Laboratoire G-SCOP, Grenoble, France

Abstract. A graph G is said to be a *Seymour graph* if for any edge set F there exist $|F|$ pairwise disjoint cuts each containing exactly one element of F, provided for every circuit C of G the necessary condition $|C \cap F| \leq |C \setminus F|$ is satisfied. Seymour graphs behave well with respect to some integer programs including multiflow problems, or more generally odd cut packings, and are closely related to matching theory.

A first coNP characterization of Seymour graphs has been shown by Ageev, Kostochka and Szigeti [1], the recognition problem has been solved in a particular case by Gerards [2], and the related cut packing problem has been solved in the corresponding special cases. In this article *we show a new, minor-producing operation that keeps this property, and prove excluded minor characterizations of Seymour graphs: the operation is the contraction of full stars, or of odd circuits.* This sharpens the previous results, providing at the same time a simpler and self-contained algorithmic proof of the existing characterizations as well, still using methods of matching theory and its generalizations.

1 Introduction

Graphs. In this paper graphs are undirected and may have loops and multiple edges. Let $G = (V, E)$ be a graph. *Shrinking* $X \subseteq V$ means the identification of the vertices in X, keeping all the edges incident to X; the result will be denoted by G/X. The *deletion* and *contraction* of an edge $e \in E$ are the usual operations (the latter is the same as shrinking the endpoints of the edge), as well as the deletion of a vertex which means the deletion of the vertex together with all the edges incident to it. We will use the notation $G - e$, G/e for the deletion, respectively contraction of edge e, and $G - v$ for the deletion of vertex v. The vertex-set, edge-set of the graph G will be denoted by $V(G)$, $E(G)$, whereas for $X \subseteq V(G)$, $\delta(X)$ will denote the *cut* induced by X that is the set of edges with exactly one endpoint in X, $E(X)$ the set of *induced* edges, that is those that have both of their endpoints in X, $I(X) = \delta(X) \cup E(X)$ and $N(X)$ the set of neighbors of X.

A graph $H = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. If $H = G(X) := (X, E(X))$ for some $X \subseteq V$, then H is called an *induced* subgraph of G (induced by X). If $\hat{F}$ is a subgraph of $\hat{G} = G/Y$ then the corresponding subgraph of G will be denoted by F.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 1–13, 2011.

Cut Packings. A family of subsets of a set S is a *packing* if the sets are disjoint, and a 2-*packing* if every $s \in S$ is contained in at most two members of the family.

Let $F \subseteq E$. A *complete packing of cuts* for (G, F) is a family of $|F|$ pairwise edge-disjoint cuts, each containing an element of F. An obvious necessary condition for the existence of a complete packing of cuts for (G, F) is that F is a *join*, that is, for every circuit C of G, $|C \cap F| \leq |C \setminus F|$. Indeed, if $\mathcal{Q}$ is a complete packing of cuts for (G, F) then for every $e \in C \cap F$ one of the cuts $Q_e \in \mathcal{Q}$ contains e, and one more edge of the circuit C which is not in F. Similarly, a *complete* 2-*packing of cuts* for (G, F) is a 2-packing of $2|F|$ cuts each containing an element of F, and the existence of a complete 2-packing of cuts for (G, F) implies that F is a join in G. These obvious facts will be used without reference all over the paper. If $\mathcal{Q}$ is a packing of cuts then we will denote by $2\mathcal{Q}$ the 2-packing of cuts obtained from $\mathcal{Q}$ by duplication.

The two graphs, the K_4 and the prism, that can be found in Figure 1 play an important role in our results.

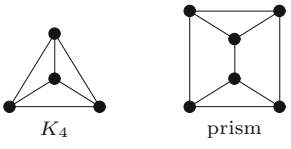


Fig. 1. The complete graph K_4 on four vertices and the prism

The graph K_4 (or the prism) with a join consisting of a perfect matching shows that a complete packing of cuts does not necessarily exist. However, by a theorem of Lovász [3] for every join F there exists a complete 2-packing of cuts for (G, F). This result (more precisely a slight sharpening) plays a key-role in the proof of all the characterizations of Seymour graphs.

Seymour graphs. The graph G is said to be a *Seymour graph* if for every join $F \subseteq E$ there exists a complete packing of cuts. Let us immediately show a condition that is sufficient for this property to fail:

Let us call a circuit C of G *tight* with respect to (G, F), if $|C \cap F| = |C \setminus F|$, that is, if the inequality $|C \cap F| \leq |C \setminus F|$ is satisfied with equality. In the above proof of "conservativeness" (from the existence of a complete packing of cuts) we have equality for tight circuits, that is in any complete packing $\mathcal{Q}$ of cuts the family $\{Q \cap C \neq \emptyset : Q \in \mathcal{Q}\}$ is a partition of C into pairs of edges $\{e, f\}$ where $e \in F$ and $f \notin F$. The following fact straightforwardly follows:

Fact 1. *Let G be a graph. If there exists a join F in G and a set of tight circuits with respect to (G, F) whose union is non-bipartite, then G is not Seymour.*

Indeed, if C is an arbitrary circuit in the union of tight circuits, and $\mathcal{Q}$ is a complete packing of cuts for (G, F), then all the edges of C are contained in

some member of $\mathcal{Q}$, by the previous remark. Therefore the nonempty members of $\{Q \cap C : Q \in \mathcal{Q}\}$ partition C. Since all the classes of this partition are even, $|C|$ is even, proving that G is bipartite.

The examples of Figure 1 show non-bipartite graphs where every edge belongs to a tight circuit for an arbitrary perfect matching, and hence these graphs are not Seymour graphs.

T-joins. Matchings and shortest paths between any two vertices of a graph are simple examples of joins (as well as arbitrary T-joins, see below.) Moreover it can be readily seen that complete packings of cuts in the dual graph (or matroid) correspond to circuits, each containing exactly one element of F, that is, to paths, each joining the endpoints of one element of F. They also occur in the dual of matching problems or of the Chinese Postman problem.

The latter has a general setting containing also matchings, planar multiflows and where the main ideas of matching theory still work: T-joins. Let $G = (V, E)$ be a connected graph and $T \subseteq V$, where $|T|$ is even. A subset $F \subseteq E$ is called a *T-join* if the set of vertices having an odd number of incident edges from F is T. For any subset $X \subseteq V$, the cut $\delta(X)$ is called a *T-cut* if $|X \cap T|$ is odd. Since in every graph the number of vertices of odd degree is even, each T-join must have at least one edge in common with each T-cut. Hence, if we denote by $\nu(G, T)$ the maximum number of edge disjoint T-cuts and by $\tau(G, T)$, the minimum cardinality of a T-join in G, then $\nu(G, T) \leq \tau(G, T)$. The example $G = K_4$ and $T = V(G)$ shows that this inequality can be strict.

The usual definition of a Seymour graph is that the equality $\nu(G, T) = \tau(G, T)$ holds for all subsets $T \subseteq V(G)$ of even cardinality. Indeed, this is equivalent to the above definition since every minimum T-join F, $|F| = \tau(G, T)$ is a join (Guan's lemma [6]) and a complete packing of cuts for (G, F) is a family of $|F| = \tau(G, T)$ disjoint T-cuts. Conversely, every join F is a minimum T-join where T is the set of vertices incident to an odd number of edges of F, and $|F| = \tau(G, T) = \nu(G, T)$ implies that a family of $\nu(G, T)$ disjoint T-cuts is a complete packing of cuts for (G, F). This reformulation is well-known and has been exploited already in Seymour's article [9].

Several particular cases of Seymour graphs have been exhibited by Seymour [8] [9], Gerards [2], Szigeti [10] while the existence of complete packing of cuts in graphs has been proved to be NP-hard [5].

Odd K_4 and odd prism. We will work with different types of subdivisions of the K_4 and the prism. Let us emphasize that in any subdivision of the K_4 or the prism every vertex is of degree 2 or 3.

A graph G is called an *odd K_4* if it is a subdivision of the K_4 such that each circuit bounding a face of G has an odd length. A graph G is an *odd prism* if it is a subdivision of the prism such that each circuit bounding a triangular face of G has an odd length while each circuit bounding a quadrangular face has an even length. (See Figure 2(a).)

A subdivision of a graph G is said to be *even* if the number of new vertices inserted in every edge of G is even (possibly zero). (See Figure 2(b).) Analogously,

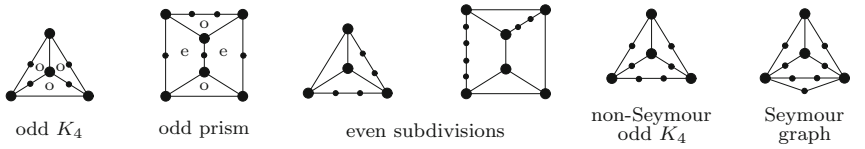


Fig. 2. (a) An odd K_4 and an odd prism, (b) even subdivisions of the K_4 and the prism, (c) a Seymour graph containing a non-Seymour odd K_4

a subdivision of a graph G is said to be *odd* if the number of new vertices inserted in every edge of G is odd. An even subdivision of the K_4 (respectively of the prism) is clearly an odd K_4 (respectively an odd prism).

Subclasses of Seymour graphs. It is well known that Seymour graphs include bipartite graphs (Seymour [8]) and series-parallel graphs (Seymour [9]). For better comparison let us mention this latter class as the family of graphs not containing any subdivision of the K_4. Gerards [2] generalized the previous results by proving that graphs not containing odd K_4 or odd prism subgraphs are Seymour. Figure 2(c) shows a subdivision of the K_4 which is not a Seymour graph but adding a new vertex joined to two old ones, or replacing the bottom path of length three by just one edge we obtain a Seymour graph. This shows that the property is not inherited to minors.

A further restriction of the excluded minors (and thus a generalization of the sufficient condition) is to exclude only the non-Seymour subdivisions of the K_4 and the prism. The list of these was provided by [10] and served as an important preliminary of [1]. Szigeti's theorem [10] stated that a graph that does not contain a non-Seymour subdivision of the K_4 or of the prism as subgraph, is a Seymour graph. This theorem generalizes all the previous results, but the sufficient condition it provides is still not necessary: the second graph on Figure 2(c) contains a non-Seymour subdivision of the K_4 (even as an induced subgraph) but is Seymour.

Continuing on this road, excluding only even subdivisions of the K_4 and the prism is already a necessary condition, that is, all Seymour graphs are contained in the defined class of graphs. Indeed, any perfect matching of an even subdivision of the K_4 is a join in the graph and there exists no complete packing of cuts. Similarly for the prism.

The main contribution of the present work is an excluded minor characterization of Seymour graphs. In Section 2 we state the main results including also the preliminaries. In Section 3 we provide a self-contained proof of the results.

2 Results

Previous characterization of Seymour graphs. The following result (conjectured by Sebő in 1992, proved by Ageev, Kostochka & Szigeti [1]), places the Seymour property in co-NP and implies all the above inclusions.

Theorem 1. *The following statements are equivalent for a graph G:*

(1) G is not a Seymour graph,
(2) G has a join F and a set of tight circuits whose union is non-bipartite,
(3) G has a join F and two tight circuits whose union forms an odd K_4 or an odd prism.

This theorem has the drawback that it depends on the join F, it is not suitable for testing the property, and heavy for algorithmic handling. A reason can be captured in the complexity of the coNP characterization: it uses a subgraph and a set F while we provide here characterizations by property keeping minor-producing operations. The main result of this paper avoids these formal drawbacks, leads to a simpler proof and is trying to pave the way to the recognition. We will exhibit several characterizations, some of which play mainly the role of a bridge in the proof, but may also be useful for studying the complexity of the problem in the future. For this we have to introduce some notions:

Factor-contraction. A graph G is called *factor-critical* if the graph $G - v$ has a perfect matching for any vertex v of G. Note that a vertex and an odd circuit is factor-critical. The following result due to Lovász [4], that provides a characterization of factor-critical graphs, will play an important role in the main result of this paper.

Fact 2. *A graph G is factor-critical if and only if a single vertex can be obtained from G by a series of odd circuit contractions.*

A *factor-contraction* is the contraction of $I(X)$ (all edges incident to a vertex set X), where $G(X)$ is factor-critical. There is a rich appearence of such sets X in the T-join structure [7], which is "robust" whith respect to the contraction of $I(X)$. We will denote the result of this operation by G^X. If $X = \{v\}$, $v \in V$, then we will write G^v, and we call this operation a *star-contraction*, the contraction of the full star of v. (A *star* is a graph consisting of edges all incident to a given vertex. A star is called *full*, if it contains all edges of G incident to v.)

We will apply the following lemma that makes possible factor-contractions unless the graph is bicritical. The first part of this statement is part of a result in [7]. The second part is much simpler and is implicit in [9].

Lemma 1. *Let $G = (V, E)$ be a graph, $F \subseteq E$ a join, and $x_0 \in V$ arbitrary.*

(1) For $F \neq \emptyset$, there exists a complete 2-packing $\{\delta(X) : X \in \mathcal{C}\}$ of cuts for (G, F) and $C \in \mathcal{C}$ so that
  (a) $G(C)$ is factor-critical,
  (b) $\{c\} \in \mathcal{C}$ for all $c \in C$ (if $|C| = 1$, then C is contained twice in $\mathcal{C}$) and
  (c) none of the members of $\mathcal{C}$ contain x_0.
(2) If there exists a complete packing of cuts for (G, F) then there is one containing a full star different of $\delta(x_0)$.

Stoc-minor. A factor-contraction can be decomposed into two operations both of which are nontrivial if $|X| > 1$: the contraction of (the edges induced by) X, and the contraction of the edges in $\delta(X)$. After the contraction of X we just have to contract a star, which thus keeps the Seymour property.

But does the contraction of a factor-critical graph also keep the property? If yes, then in particular, the contraction of an odd circuit also keeps it! By Fact 2, the contraction of the edges of a factor-critical graph is equivalent to a succession of contractions of odd circuits, in case of a yes answer, two very simple and basic operations would be sufficient to handle Seymour graphs: star- and odd-circuit-contraction. This was a raised and forgotten open question in [10].

The main novelty of the present work is that **indeed, these two operations keep the Seymour property**. This refined approach simplifies the results and their proofs of the known characterizations as well. However, we were not able to prove it directly without involving essentially the entire proof.

We will say that a graph G' is the *stoc-minor* of G if it arises from G by a series of star and odd circuit contractions. Stoc-minors would generate though an immediate simplification: prisms disappear! Indeed, the K_4 is a stoc-minor of the prism.

Biprism. The *biprism* is obtained from the prism by subdividing each edge that connects the triangles by a vertex. The set X of the three new vertices is called the *separator* of the biprism H. Note that $H - X$ has two connected components, and both are triangles. These will be called the two *sides* of the biprism. For a subdivided biprism the two sides include the new vertices of the subdivisions connected to the triangles on each side. We mention that for a subdivided biprism the underlying biprism and the two sides are not necessarily uniquely determined, so whenever necessary, we will tell more about the subdivision we are speaking about.

For $t \in \{2, 3\}$, a *t-star* is a star containing t edges. We will also deal with the families of the subgraphs L of G that are even or odd subdivisions of a t-star. The set of the t vertices of degree one in L is called the *base* of L. For a subgraph H of G, a vertex-set $U \subset V(H)$ is called a *t-core* of H if $|\delta_H(U)| = t$ and $I_H(U)$ is an even subdivision of a t-star. A *core* is either a 2-core or a 3-core.

Obstructions. We will say that G contains a *K-obstruction*, where K is the K_4, the prism or the biprism if there exists a subgraph H of G with a subpartition $\mathcal{U}$ of $V(H)$ satisfying the following properties:

- H is a subdivision of K.
- For every $U \in \mathcal{U}$ the graph $I_H(U)$ is an even subdivision of a star.
- Any path of G joining two vertices $u_1 \in U_1 \in \mathcal{U}$ and $u_2 \in V(H) - U_1$ has length at least 2 if u_2 is of degree 3 in H and it has length at least 3 if $u_2 \in U_2 \in \mathcal{U}$.

– Shrinking each $I(U)$ in G (they are disjoint because of the previous condition) we get an even subdivision of K, and for the biprism we also require that there is no edge of G between vertices on different sides. (Meaning that among the possible choices of the separator, there is one satisfying this.)

We will say that $(H,\mathcal{U})$ is an *obstruction* of G. Note that if K is the K_4 or the prism and a subgraph H of G with subpartition $\mathcal{U}$ empty is a K-obstruction, then H is simply an even subdivision of K. We mention that the graph of Figure 2(c) does not contain a K_4-obstruction because the distances of the vertices of degree three are at most two in G.

Let us explain the definition of obstructions with other words, using only contractions and not using distances (see Figure 3):

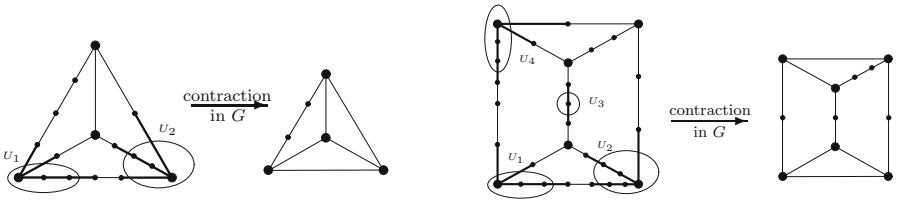


Fig. 3. (a) A K_4-obstruction and (b) a biprism-obstruction

An odd K_4 subgraph H of G with a set $\mathcal{U}$ of disjoint 3-cores U_i of H is a K_4-obstruction if after contracting in G the set of edges of G having at least one end-vertex in one of U_i's (that is $I_G(U_i)$ for $U_i \in \mathcal{U}$), H tranforms into an even subdivision of the K_4.

An odd prism subgraph H of G with a set $\mathcal{U}$ of disjoint 2 or 3-cores U_i of H is a prism-obstruction (respectively a biprism-obstruction) if after contracting in G the set of edges of G having at least one end-vertex in one of U_i's (that is $I_G(U_i)$ for $U_i \in \mathcal{U}$), H tranforms into H', an even subdivision of the prism (respectively of the biprism; and in this latter case with the further property that for a convenient separator X of the underlying biprism no edge of G connects the two connected components of $H' - X$). We mention that 2-cores are necessary because of the separator of the biprism.

Let us continue with three results on obstructions. The proofs are omitted due to space limitation.

Proposition 1. *Let $G = (V, E)$ be a graph, $A \subset V$ and $\hat{G} = G/I(A)$. Suppose that $\hat{G}$ contains a K-obstruction $(\hat{H},\hat{\mathcal{U}})$. Let H be the subgraph of G defined by the edge set of $\hat{H}$ and $T = V(H) \cap N(A)$.*

(a) If $|T| = 2$ and $G(A \cup T)$ contains an even path between the vertices of T, or

(b) if $|T| = 3$ and $G(A \cup T)$ contains an even or odd subdivision of a 3-star with base T,

then G also contains a K-obstruction.

Proposition 2. *If G contains an obstruction $(H, \emptyset)$, then G has a join F and two tight circuits whose union is H and $d_F(v) = 1$ if $d_H(v) = 3$.*

Lemma 2. *Let us suppose that the subgraph H and the subpartition $\mathcal{U}$ form an obstruction of G so that $|V(H)| + \sum_{U \in \mathcal{U}} |U|$ is minimum. Let $U \in \mathcal{U}$ be a 3-core. Then there exists no edge of G between a vertex in $N_H(U)$ and a vertex of U that are not neighbors in H.*

The next definition we need serves purely technical purposes. It is in itself not very interesting, but it is useful in the proof, and makes a bridge between the new generation of the results and the previous ones.

Visibly non-Seymour. We will say that the graph G is *visibly non-Seymour* (shortly VNS) if it has a subgraph H containing an edge set F, so that

(a) H is the union of two tight circuits with respect to (G, F),
(b) H is non-bipartite,
(c) the maximum degree is 3 in H,
(d) there exists a complete 2-packing of cuts for (G, F), which contains all the stars in G of the vertices of degree 3 in H (with multiplicity at least 1).

Matching-covered graphs. We will need some definitions and results from matching theory. A connected graph is called *matching-covered* if each edge belongs to a perfect matching. A connected graph G is called *bicritical* if the graph $G - \{u, v\}$ has a perfect matching for any two different vertices u and v of G; in other worlds: $G - u$ is factor-critical for any vertex u of G. A bicritical graph is *non-trivial* if $|V(G)| > 2$. Note that a bicritical graph is matching-covered. Adding an edge between the vertices of a bicritical graph it remains bicritical, and therefore a graph has a bicritical subgraph if and only if it has a bicritical induced subgraph. The graph K_2^3 is defined to be a set of three parallel edges.

The following two similar results on matching-covered graphs will be crucial for us. Fact 3 is due to Lovász and Plummer [4]. We mention that none of them implies the other one.

Fact 3. *Every non-bipartite matching-covered graph contains an even subdivision of the K_4 or the prism.*

Lemma 3. *Every 3-star of a matching-covered graph belongs to an even subdivision of the K_4 or the K_2^3.*

A similar observation on odd prisms will also be applied.

Fact 4. *Every 3-star of an odd prism belongs to an even or odd subdivision of the K_2^3.*

New characterization of Seymour graphs. We squeeze the main results of this paper into one theorem:

Theorem 2. *The following statements are equivalent for the graph G:*

(i) G *is not a Seymour graph,*
(ii) G *can be factor-contracted to a graph that contains a non-trivial bicritical subgraph,*
(iii) G *can be factor-contracted to a graph that contains an even subdivision of the* K_4 *or of the prism as a subgraph,*
(iv) G *contains an obstruction,*
(v) G *is a visibly non-Seymour graph,*
(vi) G *has a stoc-minor containing an even subdivision of the* K_4 *as a subgraph.*

Note that the prism disappeared! A tempting additional equivalent statement would be that G can be star-contracted to a graph that contains a bicritical graph on at least four vertices as a subgraph. This is not true as the biprism shows: it is not Seymour but contracting any of the stars it becomes Seymour. Yet it does not contain a bicritical subgraph.

3 Proof of Theorem 2

(i) implies (ii): Let $G = (V, E)$ be a minimal counter-example that is

(a) G is not a Seymour graph,
(b) every factor-contraction of G is a Seymour graph and
(c) G contains no non-trivial bicritical subgraph.

By (a), there exists a non-empty join F so that no complete packing of cuts exists for (G, F). The following lemma contradicts (c).

Lemma 4. $G' := G(V(F))$ *is a non-trivial bicritical graph.*

Proof. First we show that

no star is contained twice in a complete 2-packing $\mathcal{Q}$ of cuts for (G, F). $(*)$

To see this suppose to the contrary that $2\delta(v) \subseteq \mathcal{Q}$. Let us contract the full star of v. Then $F^v := F \setminus \delta(v)$ is a join in G^v of size $|F| - 1$ because $\mathcal{Q} - 2\delta(v)$ is a complete 2-packing of cuts for (G^v, F^v). Since the graph G^v is obtained from G by a factor-contration, we have by (b) that G^v is a Seymour graph and hence there exists a complete packing $\mathcal{Q}'$ of cuts for (G^v, F^v) and then $\mathcal{Q}' \cup \delta(v)$ is a complete packing of cuts for (G, F), which is a contradiction.

Let x_0 be an arbitrary vertex of F. Let $\mathcal{Q}$ and $C \in \mathcal{Q}$ be the sets provided by (1) of Lemma 1. We recall that $G(C)$ is factor-critical and $C \subseteq V(F) - x_0$. In fact,

$$C = V(F) - x_0. \qquad (**)$$

To see this suppose to the contrary that $C \subset V(F) - x_0$. Then, since $\delta(C)$ contains only one edge of F, the set $F^C := F \setminus I(C)$ is non-empty. This edge set

F^C is a join in G^C, since $\mathcal{Q} \setminus (\{\delta(C)\} \cup \{\delta(c) : c \in C\})$ is a complete 2-packing of cuts for (G^C, F^C). By (b), G^C is a Seymour graph and hence there exists a complete packing $\mathcal{Q}^C$ of cuts for (G^C, F^C). By (2) of Lemma 1, $\delta(v) \in \mathcal{Q}^C$ for some $v \in V(G) \setminus (C \cup N(C))$. Then $(2\mathcal{Q}^C) \cup (\{\delta(C)\} \cup \{\delta(c) : c \in C\})$ is a complete 2-packing of cuts for (G, F), which is a contradiction by (*).

It follows, by (*), that $|C| \neq 1$ and by (**), that $G' - x_0 = G(C)$ is factor-critical for every $x_0 \in V(G')$, that is G' is a non-trivial bicritical graph. □

(ii) implies (iii): By Fact 3, a non-trivial bicritical graph contains an even subdivision of the K_4 or the prism.

(iii) implies (iv): Let G satisfy (iii), that is, G can be factor-contracted to a graph $\hat{G}$ that contains an even subdivision H of the K_4 or the prism. Then $(H, \emptyset)$ is an obstruction of $\hat{G}$. To show that G satisfies (iv), that is G itself contains an obstruction, we prove the following lemma and then we are done by induction:

Lemma 5. *If G^C ($C \subseteq V$, $G(C)$ is factor-critical) contains an obstruction, then so does G.*

Proof. Let $\hat{G}_1 := G^C$, $(\hat{H}_1, \hat{\mathcal{U}})$ an obstruction of $\hat{G}_1$, H_1 the subgraph of G defined by the edge set of $\hat{H}_1$ and $T := \{v_1, \ldots, v_l\} = V(H_1) \cap N(C)$. Since the vertices of $\hat{H}_1$ are of degree 2 or 3, we have that $l \leq 3$.

Case 1. If $l \leq 1$, then $(H_1, \mathcal{U})$ is an obstruction of G, and we are done.

Thus we may suppose without loss of generality that $l \geq 2$. Let $\hat{G}_2$ be the graph obtained from G by contracting $V - C$. Since $v_i \in N(C)$, we can choose a neighbor u_i of v_i in C for all $i = 1, \ldots, l$. Since $G(C)$ is factor-critical, $G(C) - u_i$ contains a perfect matching and hence $\hat{G}_2$ has a perfect matching M_i containing $v_i u_i$. Then the subgraph $\hat{S}$ of $\hat{G}_2$ induced by the edge set $\bigcup_1^l M_i$ is matching-covered.

Case 2. If $l = 2$, then S is an even path in $G(C \cup T)$ and by Proposition 1(a), G contains an obstruction, and we are done.

Thus from now on we suppose that $l = 3$. By Lemma 3, there exists in $\hat{S}$ an even subdivision $\hat{H}_2$ either of the K_4 or of the K_2^3 containing the three edges u_1v_1, u_2v_2 and u_3v_3.

Case 3. If $\hat{H}_2$ is an even subdivision of the K_2^3, then H_2 is an even subdivision of a 3-star with base T in $G(C \cup T)$ and by Proposition 1(b), we are done.

So we suppose that $\hat{H}_2$ is an even subdivision of K_4, that is $(\hat{H}_2, \emptyset)$ is a K_4-obstruction in $\hat{S}$.

Case 4. If $\hat{H}_1$ is an odd prism of $\hat{G}_1$, then by Fact 4, there exists an even or odd subdivision $\hat{L}$ of the K_2^3 in $\hat{G}_1$, so L is an even or odd subdivision of the 3-star in H_1 with base T in $G((V - C - N(C)) \cup T)$ and we are done again by Proposition 1(b).

Case 5. Finally, if $\hat{H}_1$ is an odd K_4 in $\hat{G}_1$ and $\hat{H}_2$ is an even subdivision of the K_4 in $\hat{G}_2$, then $H_1 \cup H_2$ is an odd prism of G. If the contracted vertex of $\hat{G}_1$

belongs to one of the cores $\hat{U}_i$ say to $\hat{U}_j$, then let $U_i := \hat{U}_i$ for $i \neq j$ and U_j is deleted, otherwise let $U_i := \hat{U}_i$ for $1 \leq i \leq k$. Then $(H_1 \cup H_2, \mathcal{U})$ is an obstruction of G because after contracting the sets $U_i \in \mathcal{U}$ we get an even subdivision of the prism or the biprism. □

(iv) implies (v): Let $(H,\mathcal{U})$ be an obstruction of G, $G' := G/\cup_1^k I_G(U_i)$ and $H' := H/\cup_1^k I_G(U_i)$. By Proposition 2, there exist a join F' of G' and two tight circuits C_1' and C_2' whose union is H' and for each 3-core U_i exactly one edge e_i of F' is incident to the vertex u_i that corresponds to U_i in G'. For each 3-core U_i there is a unique edge $v_iw_i \in \delta_H(U_i)$ such that $v_i \in U_i, w_i \in V - U_i$ and e_i is incident to w_i. For each 2-core U_i, let w_iv_i be one of the two edges of the cut $\delta_H(U_i)$ with $v_i \in U_i$. For each core U_i, let F_i be the unique perfect matching of $H(U_i) - v_i$. Let $F := F' \cup_1^k (F_i + v_iw_i)$.

By Lemma 1, there exists a complete 2-packing $\mathcal{Q}_0'$ of cuts for (G', F'). Let $\mathcal{Q}_0$ be the corresponding complete 2-packing of cuts for (G, F'). Let $\mathcal{Q}_i := \{v : v \in U_i\} \cup \{U_i\}$ and $\mathcal{Q} := \cup_0^k \mathcal{Q}_i$. Then $\mathcal{Q}$ is a complete 2-packing of cuts for (G, F) hence F is a join of G.

Moreover, C_1' and C_2' correspond to two tight circuits of G whose union is H. Thus G is VNS.

(v) implies (i): Let H be a subgraph of G, F an edge set in H and $\mathcal{Q}$ a complete 2-packing of cuts for (G, F) that show that G is VNS. By (d), F is a join of G, by (a), H is the union of two tight circuits with respect to (G, F), and by (b), H is non-bipartite, so by Fact 1, G is not a Seymour graph.

(iii) implies (vi): By Fact 2, a factor-contraction can be replaced by a series of contractions of odd circuits and then the contraction of a star. Moreover, we can contract an odd circuit in an even subdivion of the prism to get an even subdivision of the K_4. Hence, if G can be factor-contracted to a graph that contains an even subdivision of the K_4 or of the prism as a subgraph, then G has a stoc-minor containing an even subdivision of the K_4 as a subgraph.

(vi) implies (i): Let $\hat{G}$ be a stoc-minor of G that contains an even subdivision H of the K_4. Then any perfect matching of H is a join of $\hat{G}$, and H is the union of two tight circuits with respect to $(\hat{G}, F)$, so by Fact 2, $\hat{G}$ is not a Seymour graph, that is $\hat{G}$ satisfies (i). To show that G satisfies (i) we prove the following lemma and then we are done by induction:

Lemma 6. *Let C be a full star or an odd circuit. If G/C satisfies (i), then so does G.*

Proof. If C is a full star, then since G/C satisfies (i), it satisfies **(iii)**. A full star contraction is a factor-contraction, so G also satisfies **(iii)** and hence (i).

From now on C is an odd circuit. Since G/C satisfies (i), it satisfies **(iv)**, that is G/C contains a K-obstruction $(\hat{H},\mathcal{U})$ where K is the K_4, the prism or the biprism. Take a minimal one as in Lemma 2.

Case 1: If $|V(C) \cap V(H)| \leq 2$ and c is not in a separator of the biprism. If $c \in V \setminus \cup N_H(U_i)$, then the obstruction $\hat{H}$ can be extended by the even part of C, that is G also satisfies **(iv)** and hence (i). Otherwise, $c \in N_H(U_i)$ for some i, that is H contains an edge cy so that $y \in U_i$.

If there exists no edge of G between U_i and the even path C_2 of the cycle C between c_1 and c_2 then the obstruction $\hat{H}$ can be extended by C_2, that is G also satisfies **(iv)** and hence (i). Otherwise, take that edge xc_3 of G with $x \in U_i$ for which c_3 is on C_2 and the distance d of c_2 and c_3 on C_2 is as small as possible. By Lemma 2, $x = y$. If d is even then for the edge yc_3 we are in the above case. If d is odd, then changing H as shown on Figure 4, where Q is an odd path in H, and deleting U from $\mathcal{U}$, we get a K_4-obstruction.

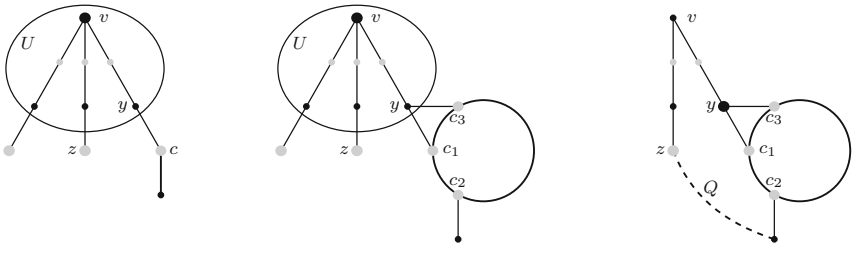


Fig. 4. A new K_4-obstruction

Case 2: If $|V(C) \cap V(H)| = 2$ and c is in a separator of the biprism. Let a and b be the two vertices of C that are incident to the two edges of H incident to c. Then a and b partition C into an odd path C_1 and an even path C_2. If $a = b$, then $(\hat{H},\mathcal{U})$ remains a biprism-obstruction in G, so G satisfies **(iv)** and hence G also satisfies (i). If $a \neq b$, then let H' be obtained from $\hat{H}$ by adding C_1 and deleting the path between v_5 and v_6 defined in Figure 5. Let $\mathcal{U}'$ be obtained from $\mathcal{U}$ by deleting those cores that correspond to inner vertices of the deleted path. Then $(H',\mathcal{U}')$ is a K_4-obstruction in G, so G satisfies **(iv)** and hence G also satisfies (i). (See Figure 5)

Case 3: If $|V(C) \cap V(H)| = 3$. Then since G/C satisfies (iv), it satisfies **(v)** with the same $\hat{H}$. By (d) of VNS for c, there exists exactly one edge e in F incident to c and both tight circuits contains e. Let u be the end vertex of e in C, M a perfect matching of $C - u$, $F^* = F \cup M$ and $H^* = H \cup C$. The vertices of H partition C into 3 paths. If exatly one of them is of odd length, then delete from H^*, F^* and C that odd path. Let $\mathcal{Q}^* := \mathcal{Q} \setminus \delta(c) \cup \{\delta(x) : x \in C'\}$. Then H^* satisfies (a), (b), (c) and (d) of VNS, so G satisfies **(v)** and hence (i). □

In order to convert this proof into an algorithm for either finding the complete packing or certifying that the graph is not Seymour no more ideas are needed. However, the size constraints of this volume oblige us to postpone this for the full journal version of this paper. The problem of recognizing Seymour graphs is still open.

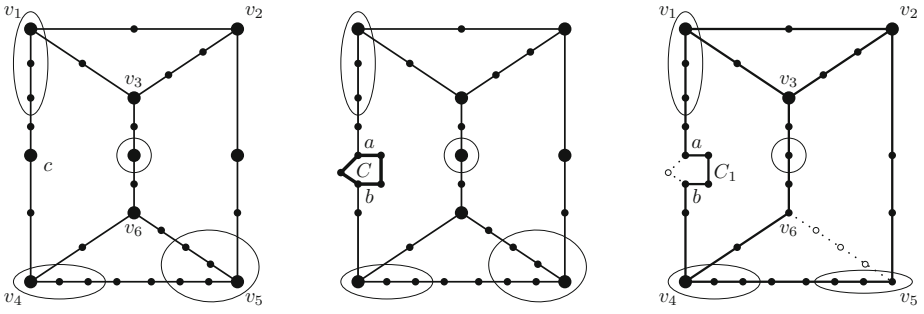


Fig. 5. Finding a K_4-obstruction

References

1. Ageev, A.A., Kostochka, A.V., Szigeti, Z.: A characterization of Seymour graphs. J. Graph Theory 24, 357–364 (1997)
2. Gerards, A.M.H.: On shortest T-joins and packing T-cuts. J. Combin. Theory Ser. B 5, 73–82 (1992)
3. Lovász, L.: 2-matchings and 2-covers of hypergraphs. Acta. Math. Sci. Hungar. 26, 433–444 (1975)
4. Lovász, L., Plummer, M.D.: Matching Theory, Akadémiai Kiadó, Budapest (1986)
5. Middendorf, M., Pfeiffer, F.: On the complexity of the edge-disjoint paths problem. Combinatorica 13(1), 97–108 (1993)
6. Guan, M.G.: Graphic programming using odd and even points. Chinese Mathematics 1, 273–277 (1962)
7. Sebö, A.: o, Undirected distances and the postman structure of graphs. J. Combin. Theory Ser. B 49, 10–39 (1990)
8. Seymour, P.D.: The matroids with the max-flow min-cut property. J. Combin. Theory Ser. B 23, 189–222 (1977)
9. Seymour, P.D.: On odd cuts and plane multicommodity flows. Proc. London Math. Soc. Ser. 42(3), 178–192 (1981)
10. Szigeti, Z.: On Seymour graphs, Technical Report No. 93803, Institute for Operations Research, Universität Bonn (1993)

Complexity Analyses of Bienstock–Zuckerberg and Lasserre Relaxations on the Matching and Stable Set Polytopes*

Yu Hin Au and Levent Tunçel

Department of Combinatorics and Optimization, Faculty of Mathematics,
University of Waterloo, Waterloo, Ontario,
N2L 3G1 Canada

Abstract. Many hierarchies of lift-and-project relaxations for 0,1 integer programs have been proposed, two of the most recent and strongest being those by Lasserre in 2001, and Bienstock and Zuckerberg in 2004. We prove that, on the LP relaxation of the matching polytope of the complete graph on $(2n+1)$ vertices defined by the nonnegativity and degree constraints, the Bienstock–Zuckerberg operator (even with positive semidefiniteness constraints) requires $\Theta(\sqrt{n})$ rounds to reach the integral polytope, while the Lasserre operator requires $\Theta(n)$ rounds. We also prove that Bienstock–Zuckerberg operator, without the positive semidefiniteness constraint requires approximately $n/2$ rounds to reach the stable set polytope of the n-clique, if we start with the fractional stable set polytope. As a by-product of our work, we consider a significantly strengthened version of Sherali–Adams operator and a strengthened version of Bienstock–Zuckerberg operator. Most of our results also apply to these stronger operators.

Keywords: matching polytope, lift-and-project methods, integer programming, semidefinite programming, convex relaxations.

1 Introduction

Given a polytope $P \subseteq [0,1]^n$, we are interested in $P_I := \text{conv}\{P \cap \{0,1\}^n\}$, the *integer hull* of P. While it is impossible to efficiently find a description of P_I for a general P (unless $\mathcal{P} = \mathcal{NP}$), we may use properties that we know are satisfied by points in P_I to derive inequalities that are valid for P_I but not P.

Lift-and-Project methods provide a systematic way to generate a sequence of convex relaxations converging to the integer hull P_I. These methods have a special place in optimization as they lie at the intersection of combinatorial optimization and convex analysis (this goes back to work by Balas and others in the late 1960s and the early 1970s, see for instance Balas [Bal98] and the references therein). Some of the most attractive features of these methods are:

* This research was supported in part by an OGSST Scholarship, a Sinclair Scholarship, an NSERC Scholarship and NSERC Discovery Grants.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 14–26, 2011.

- Convex relaxations of P_I obtained after $O(1)$ rounds of the procedure are *tractable* provided P is tractable (here tractable means that the underlying linear optimization problem is polynomial-time solvable).
- Many of these methods use *lifted* (higher dimensional) representations for the relaxations. Such representations sometimes allow compact (polynomial size in the input) convex representations of exponentially many facets.
- Most of these methods allow addition of positive semidefiniteness constraints in the lifted-space. This feature can make the relaxations much stronger in some cases, without sacrificing polynomial-time solvability. Moreover, these semidefiniteness constraints can represent an uncountable family of defining linear inequalities, such as those of the theta body of a graph.
- Systematic generation of tighter and tighter relaxations converging to P_I in at most n rounds makes the strongest of these methods good candidates for utilization in generating polynomial time approximation algorithms for hard problems, or for proving large integrality gaps (hence providing a negative result about approximability in the underlying hierarchy).

In the last two decades, many lift-and-project operators have been proposed (see [SA90], [LS91], [BCC93], [Las01] and [BZ04]), and have been applied to various discrete optimization problems. For instance, many families of facets of the stable set polytope of graphs are shown to be easily generated by these procedures [LS91] [LT03]. Also studied are their performances on set covering [BZ04], TSP relaxations [CD01] [Che05], max-cut [Lau02], and so on. For general properties of these operators and some comparisons among them, see [GT01], [HT08] and [Lau03].

In this paper, we focus on the strongest of the existing operators. Analyzing the behaviour of these strong operators on fundamental combinatorial optimization problems such as matching and the stable set problem, improves our understanding of these operators and their limitations. This in turn provides future research directions for further improvements of these hierarchies and related algorithms as well as the design and discovery of new ones.

Two of the strongest operators known to date are Las by Lasserre and BZ_+ by Bienstock and Zuckerberg. We are interested in these strongest operators because they provide the strongest tractable relaxations obtained this way. On the other hand, if we want to prove that some combinatorial optimization problem is difficult to attack by lift-and-project methods, then we would hope to establish them on the strongest existing hierarchy for the strongest negative results. For example, some of the known non-approximability results on vertex cover are based on Lovász and Schrijver's LS_+ operator ([GMPT06], [STT06]), which is known not to be the strongest. By understanding the more powerful operators (or better yet, inventing new ones), we could either obtain better approximations for vertex cover (and other hard problems), or lay the groundwork for yet stronger non-approximability results.

Our most striking findings are on the matching and stable set polytopes. Stephen and the second author [ST99] proved that LS_+ requires n rounds on the matching polytope of the $(2n + 1)$-clique, establishing the first bad

instance for LS_+ since it was proposed in 1991. Subsequently, some other lift-and-project operators have been shown to also perform poorly in this instance. For the Balas–Ceria–Cornuéjols operator [BCC93], Aguilera, Bianchi and Nasini [ABN04] showed that n^2 rounds are needed. More recently, Mathieu and Sinclair [MS09] proved that Sherali–Adams operator requires $(2n-1)$ rounds. The related question for the Lasserre operator has been open since 2001, and for the BZ_+ operator since 2003. We answer these questions as $\Theta(n)$ rounds and $\Theta(\sqrt{n})$ rounds respectively. This establishes the first example on which BZ_+ requires more than $O(1)$ rounds to reach the integer hull. For some bad instances for Lasserre's operator, see [Lau02] and [Che07]. An implication of our results is that all of these procedures become exponential time algorithms on the matching problem (assuming that they are implemented as stated). As a by-product of our analysis, we develop some new tools and modify some existing ones in the area. We also construct a very strong version of Sherali–Adams operator that we call SA_+ (there are other weaker versions of SA_+ in the recent literature called *Sherali–Adams SDP*, see [BGM10] and the references therein) and relate it to the BZ_+ operator. Also, as another by-product of our approach we strengthen the BZ, BZ_+ operators (our analyses also applies to these stronger versions). We conclude the paper by proving that the BZ operator requires approximately $n/2$ rounds to compute the stable set polytope of the n-clique.

2 Preliminaries

For convenience, we denote $\{0,1\}^n$ by $\mathcal{F}$ and the set $\{1,2,\ldots,n\}$ by $[n]$ herein. Given $x \in [0,1]^n$, let $\hat{x}$ denote the vector $\begin{pmatrix}1\\x\end{pmatrix}$ in $\mathbb{R}^{n+1}$, where the new coordinate is indexed by 0. Let e_i denote the i^{th} unit vector, and for any square matrix M let $\text{diag}(M)$ denote the vector formed by the diagonal entries of M. Next, given $P \subseteq [0,1]^n$, define the cone

$$K(P) := \left\{ \begin{pmatrix}\lambda\\ \lambda x\end{pmatrix} \in \mathbb{R} \oplus \mathbb{R}^n : \lambda \geq 0, x \in P \right\}.$$

Define $\mathcal{A} := 2^{\mathcal{F}}$. For each $x \in \mathcal{F}$, we define the vector $x^{\mathcal{A}} \in \mathbb{R}^{\mathcal{A}}$ such that

$$x^{\mathcal{A}}_{\alpha} = \begin{cases} 1 \text{ if } x \in \alpha \\ 0 \text{ otherwise.} \end{cases}$$

I.e., each coordinate of $\mathcal{A}$ can be interpreted as a subset of the vertices of the n-dimensional hypercube, and $x^{\mathcal{A}}_{\alpha} = 1$ if and only if the point x is contained in the set α. It is not hard to see that for all $x \in \mathcal{F}$ and $i \in [n]$, we have $x^{\mathcal{A}}_{\mathcal{F}} = 1$, and $x^{\mathcal{A}}_{\{y \in \mathcal{F} : y_i = 1\}} = x_i$. Another important property of $x^{\mathcal{A}}$ is that, given disjoint subsets $\alpha_1, \alpha_2, \ldots, \alpha_k \subseteq \mathcal{F}$, we know that

$$x^{\mathcal{A}}_{\alpha_1} + x^{\mathcal{A}}_{\alpha_2} + \cdots + x^{\mathcal{A}}_{\alpha_k} \leq 1, \tag{1}$$

and equality holds if $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ partitions $\mathcal{F}$. Let $\mathbb{S}^n$ denote the set of n-by-n real, symmetric matrices and $\mathbb{S}^n_+ \subset \mathbb{S}^n$ denote the set of symmetric, positive

semidefinite n-by-n matrices. For any given $x \in \mathcal{F}$, if we define $Y_{\mathcal{A}}^x := x^{\mathcal{A}}(x^{\mathcal{A}})^T$, then we know that the entries of $Y_{\mathcal{A}}^x$ have considerable structure. Most notably, the following must hold:

- $Y_{\mathcal{A}}^x \in \mathbb{S}_+^{\mathcal{A}}$; $Y_{\mathcal{A}}^x e_{\mathcal{F}} = (Y_{\mathcal{A}}^x)^T e_{\mathcal{F}} = \operatorname{diag}(Y_{\mathcal{A}}^x) = x^{\mathcal{A}}$; $Y_{\mathcal{A}}^x e_\alpha \in \left\{0, x^{\mathcal{A}}\right\}$, $\forall \alpha \in \mathcal{A}$;
- $Y_{\mathcal{A}}^x[\alpha, \beta] = 1 \iff x \in \alpha \cap \beta$
- If $\alpha_1 \cap \beta_1 = \alpha_2 \cap \beta_2$, then $Y_{\mathcal{A}}^x[\alpha_1, \beta_1] = Y_{\mathcal{A}}^x[\alpha_2, \beta_2]$.

Zuckerberg [Zuc03] showed that most of the existing lift-and-project operators can be interpreted under the common theme of placing constraints that are relaxations of the above conditions on submatrices of $Y_{\mathcal{A}}^x$. In the remainder of this section, we define the operators proposed by Lasserre [Las01] and Bienstock–Zuckerberg [BZ04]. However, it is helpful to look at a strengthened version of the Sherali–Adams' operator [SA90] first, which has an additional positive semidefiniteness constraint. (We denote the new operator by SA_+.) Our SA_+ is similar to an operator studied by Benabbas, Georgiou and Magen [BGM10] and others, even though our version is stronger. The SA_+ operator will be useful in simplifying our analysis and improving our understanding of the Bienstock–Zuckerberg operator. For ease of reference, Figure 1 provides a glimpse of the relative strengths of various known lift-and-project operators. Each arrow in the chart denotes "is refined by" (i.e. the operator that is at the head of an arrow is stronger than that at the tail). Also, operators whose labels are framed in a box are new (the operators BZ′ and BZ'_+ will be defined in Section 2.3).

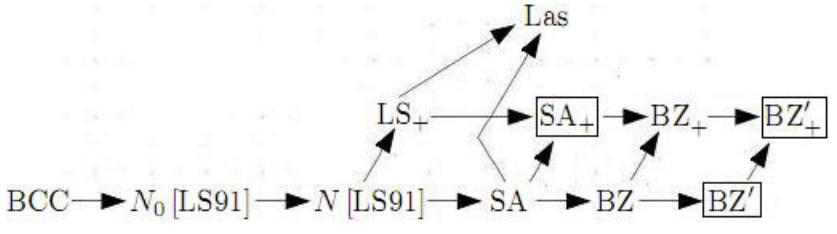


Fig. 1. A strength chart of lift-and-project operators

While all of these operators can be applied to any polytope contained in the unit hypercube (and in the Lasserre operator's case, sets defined by polynomial inequalities), we will focus our discussion on their application to lower-comprehensive polytopes (polytopes P such that $u \in P$, $0 \leq v \leq u$ implies $v \in P$), since our main objects of interest are the matching and stable set polytopes.

2.1 The SA and SA_+ Operator

Given a set of indices $S \subseteq [n]$ and $t \in \{0, 1\}$, we define

$$S|_t := \{x \in \mathcal{F} : x_i = t,\ \forall i \in S\}. \tag{2}$$

To reduce cluttering, we write $i|_t$ instead of $\{i\}|_t$. Also, for ease of reference, given any $\alpha \in \mathcal{A}$ in the form of $S|_1 \cap T|_0$ where $S, T \subseteq [n]$ are disjoint, we call S the set of *positive indices* in α, T the set of *negative indices* in α, and $|S| + |T|$ the *order* of α. Finally, for any integer $i \in [0, n]$, define $\mathcal{A}_i := \{S|_1 \cap T|_0 : S, T \subseteq [n], S \cap T = \emptyset, |S| + |T| \leq i\}$ and $\mathcal{A}_i^+ := \{S|_1 : S \subseteq [n], |S| \leq i\}$. Given a fixed integer $k \in [1, n]$, the SA^k and SA_+^k operators can be defined as follows:

1. Let $\tilde{\mathrm{SA}}^k(P)$ denote the set of matrices $Y \in \mathbb{R}^{\mathcal{A}_1^+ \times \mathcal{A}_k}$ that satisfy all of the following conditions:

(SA 1) $Y[\mathcal{F}, \mathcal{F}] = 1$;
(SA 2) $\hat{x}(Ye_\alpha) \in K(P)$ for every $\alpha \in \mathcal{A}_k$;
(SA 3) For each $S|_1 \cap T|_0 \in \mathcal{A}_{k-1}$, impose

$$Ye_{S|_1 \cap T|_0} = Ye_{S|_1 \cap T|_0 \cap j|_1} + Ye_{S|_1 \cap T|_0 \cap j|_0}, \ \forall j \in [n] \setminus (S \cup T).$$

(SA 4) For each $\alpha \in \mathcal{A}_1^+, \beta \in \mathcal{A}_k$ such that $\alpha \cap \beta = \emptyset$, impose $Y[\alpha, \beta] = 0$;
(SA 5) For every $\alpha_1, \alpha_2 \in \mathcal{A}_1^+, \beta_1, \beta_2 \in \mathcal{A}_k$ such that $\alpha_1 \cap \beta_1 = \alpha_2 \cap \beta_2$, impose $Y[\alpha_1, \beta_1] = Y[\alpha_2, \beta_2]$.

2. Let $\tilde{\mathrm{SA}}_+^k(P)$ denote the set of matrices $Y \in \mathbb{S}_+^{\mathcal{A}_k}$ that satisfies all of the following conditions:

(SA_+ 1) (SA 1), (SA 2) and (SA 3);
(SA_+ 2) For each $\alpha, \beta \in \mathcal{A}_k$ such that $\alpha \cap \beta \cap P = \emptyset$, impose $Y[\alpha, \beta] = 0$;
(SA_+ 3) For any $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathcal{A}_k$ such that $\alpha_1 \cap \beta_1 = \alpha_2 \cap \beta_2$, impose $Y[\alpha_1, \beta_1] = Y[\alpha_2, \beta_2]$.

3. Define

$$\mathrm{SA}^k(P) := \left\{ x \in \mathbb{R}^n : \exists Y \in \tilde{\mathrm{SA}}^k(P) : Ye_{\mathcal{F}} = \hat{x} \right\}$$

and

$$\mathrm{SA}_+^k(P) := \left\{ x \in \mathbb{R}^n : \exists Y \in \tilde{\mathrm{SA}}_+^k(P) : \hat{x}(Ye_{\mathcal{F}}) = \hat{x} \right\}.$$

The SA_+^k operator extends the lifted space of the SA^k operator to a set of square matrices, and imposes an additional positive semidefiniteness constraint. Moreover, SA_+^k refines the LS_+^k operator devised by Lovász and Schrijver in [LS91], which we define below. Given $P \subseteq [0, 1]^n$,

$$\mathrm{LS}_+(P) := \{x \in \mathbb{R}^n : \exists Y \in \mathbb{S}_+^{n+1}; Ye_0 = \mathrm{diag}(Y) = \hat{x}; \\ Ye_i, Y(e_0 - e_i) \in K(P), \ \forall i \in [n]\}.$$

For any integer $k \geq 1$, define $\mathrm{LS}_+^k(P) := \mathrm{LS}_+(\mathrm{LS}_+^{k-1}(P))$, where $\mathrm{LS}_+^0(P) := P$. Then we have the following:

Proposition 1. *For every polytope $P \subseteq [0, 1]^n$ and every integer $k \geq 1$, $\mathrm{SA}_+^k(P) \subseteq \mathrm{LS}_+(\mathrm{SA}_+^{k-1}(P))$.*

It follows immediately from Proposition 1 that $\mathrm{SA}_+^k(P) \subseteq \mathrm{LS}_+^k(P)$. We also remark that the condition (SA_+ 2) can be efficiently checked. For $\alpha, \beta \in \mathcal{A}_k$, $\alpha = S|_1 \cap T|_0$ and $\beta = S'|_1 \cap T'|_0$, $\alpha \cap \beta \cap P = \emptyset$ if and only if $\chi^{S \cup S'} \notin P$, since P is lower-comprehensive.

2.2 The Lasserre Operator

We next turn our attention to Lasserre's operator defined in [Las01], denoted Las^k herein. Our presentation of the operator is closer to that in [Lau03]. Given $P := \{x \in [0,1]^n : Ax \leq b\}$, and an integer $k \in [n]$,

1. Let $\tilde{\text{Las}}^k(P)$ denote the set of matrices $Y \in \mathbb{S}_+^{\mathcal{A}_{k+1}^+}$ that satisfy all of the following conditions:

(Las 1) $Y[\mathcal{F}, \mathcal{F}] = 1$;

(Las 2) For each $j \in [m]$, let A^j be the j^{th} row of A. Define the matrix $Y^j \in \mathbb{S}^{\mathcal{A}_k^+}$ such that

$$Y^j[S|_1, S'|_1] := b_j Y[S|_1, S'|_1] - \sum_{i=1}^{n} A_i^j Y[(S \cup \{i\})|_1, (S' \cup \{i\})|_1]$$

and impose $Y^j \succeq 0$.

(Las 3) For every $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathcal{A}_k^+$ such that $\alpha_1 \cap \beta_1 = \alpha_2 \cap \beta_2$, impose $Y[\alpha_1, \beta_1] = Y[\alpha_2, \beta_2]$.

2. Define

$$\text{Las}^k(P) := \left\{ x \in \mathbb{R}^n : \exists Y \in \tilde{\text{Las}}^k(P) : \hat{x}(Ye_{\mathcal{F}}) = \hat{x} \right\}.$$

In our setting, the Las-rank of a polytope P (the smallest k such that $\text{Las}^k(P) = P_I$) is equal to the Theta-rank, defined in [GPT10].

2.3 The Bienstock-Zuckerberg Operators

Finally, we look into the operators devised by Bienstock and Zuckerberg [BZ04]. One of the features of their algorithms is that they use variables in $\mathcal{A}$ that were not exploited by the operators proposed earlier, in conjunction with some new constraints. We denote their algorithms by BZ and BZ_+, but we also present modified variants called BZ' and BZ'_+. These modified algorithms have the advantage of being stronger; moreover, they are simpler to present. We discuss this in more detail after stating the algorithm.

Suppose we are given $P := \{x \in [0,1]^n : Ax \leq b\}$. The BZ' algorithm can be viewed as a two-step process. The first step is *refinement*. Recall that A^i is the i-th row of A. If $O \subseteq [n]$ satisfies

$$O \subseteq \text{supp}(A^i); \sum_{j \in O} A_j^i > b_i; |O| \leq k \text{ or } |O| \geq |\text{supp}(A^i)| - k,$$

for some $i \in [m]$, then we call O a *k-small obstruction*. (Here, $\text{supp}(a)$ denotes the support of a, i.e., the index set of nonzero components of a.) Let $\mathcal{O}$ denote the set of all k-small obstructions of P (or more precisely, of the system $Ax \leq b$). Notice that, for any obstruction $O \in \mathcal{O}$, and for every integral vector $x \in P$, the inequality $\sum_{i \in O} x_i \leq |O| - 1$ holds. Thus,

$$P_{\mathcal{O}} := \left\{ x \in P : \sum_{i \in O} x_i \leq |O| - 1, \ \forall O \in \mathcal{O} \right\}$$

is a relaxation of the integer hull of P that is potentially tighter than P. We call $P_{\mathcal{O}}$ the *k-refinement* of P.

The second step of the algorithm is *lifting*. Before we give the details of this step, we need another intermediate collection of sets of indices, called *walls*. We call $W \subseteq [n]$ a wall if either $W = \{i\}$ for some $i \in [n]$, or if there exist $\ell \leq k$ distinct obstructions $O_1, \ldots, O_\ell \in \mathcal{O}$ such that $W = \bigcup_{i,j\in[\ell],i\neq j}(O_i \cap O_j)$. That is, each subset of up to k obstructions generate a wall, which is the set of elements that appear in at least two of the given obstructions.

Next, we define the collection of *tiers*

$$\mathcal{S} := \left\{ S \subseteq [n] : \exists W_{i_1}, \ldots, W_{i_{k-1}} \in \mathcal{W}, S \subseteq \bigcup_{j=1}^{k-1} W_{i_j} \right\}.$$

I.e., we include a set of indices S as a tier if there exist $k-1$ walls whose union contains S. Note that the empty set and the singleton-sets are always tiers.

Finally, for any set $U \subseteq [n]$ and a nonnegative integer r, we define

$$U|_{<r} := \left\{ x \in \{0,1\}^n : \sum_{i\in U} x_i \leq r-1 \right\}. \tag{3}$$

We will see that the elements in $\mathcal{A}$ that are being generated by BZ′ all take the form $S|_1 \cap T|_0 \cap U|_{<r}$, where S, T, U are disjoint sets of indices. For a set α in this form, we let S (resp. T) denote the set of positive (resp. negative) indices of α, and define the order of α to be $|S| + |T| + |U|$. We are now ready to describe the lifting step:

1. Define $\mathcal{A}'$ to be the set consisting of the following: For each tier $S \in \mathcal{S}$ and each $T \subseteq S$ such that $|T| \leq k-1$, include the sets

$$(S \setminus T)|_1 \cap T|_0; \tag{4}$$

if $|T| < k-1$, and $U \subseteq S \setminus T$ such that $|U| + |T| > k-1$, then also include

$$(S \setminus (T \cup U))|_1 \cap T|_0 \cap U|_{<|U|-(k-1-|T|)}, \tag{5}$$

2. Let $\tilde{\mathrm{BZ}}'^k(P)$ denote the set of matrices $Y \in \mathbb{S}^{\mathcal{A}'}$ that satisfy all of the following conditions:

(BZ′ 1) $Y[\mathcal{F}, \mathcal{F}] = 1$;

(BZ′ 2) For any column x of the matrix Y,

(i) $0 \leq x_\alpha \leq x_{\mathcal{F}}$, for all $\alpha \in \mathcal{A}'$;

(ii) $\hat{x}(x) \in K(P_{\mathcal{O}})$;

(iii) $x_{i|_1} + x_{i|_0} = x_{\mathcal{F}}$ for every $i \in [n]$;

(iv) For each $\alpha \in \mathcal{A}'$ in the form of $S|_1 \cap T|_0$ impose the inequalities

$$x_{i|_1} \geq x_\alpha, \quad \forall i \in S, \tag{6}$$

$$x_{i|_0} \geq x_\alpha, \quad \forall i \in T, \tag{7}$$

$$x_\alpha + x_{(S\cup\{i\})|_1\cap(T\setminus\{i\})|_0} = x_{S|_1\cap(T\setminus\{i\})|_0}, \quad \forall i \in T, \tag{8}$$

$$\sum_{i\in S} x_{i|_1} + \sum_{i\in T} x_{i|_0} - x_\alpha \leq (|S| + |T| - 1)x_{\mathcal{F}}. \tag{9}$$

For each $\alpha \in \mathcal{A}'$ in the form $S|_1 \cap T|_0 \cap U|_{<r}$, impose the inequalities

$$x_{i|_1} \geq x_\alpha, \quad \forall i \in S, \tag{10}$$

$$x_{i|_0} \geq x_\alpha, \quad \forall i \in T, \tag{11}$$

$$\sum_{i \in U} x_{i|_0} \geq (|U| - (r-1))x_\alpha, \tag{12}$$

$$x_\alpha = x_{S|_1 \cap T|_0} - \sum_{U' \subseteq U, |U'| \geq r} x_{(S \cup U')|_1 \cap (T \cup (U \setminus U'))|_0}. \tag{13}$$

(BZ′ 3) For each pair $\alpha, \beta \in \mathcal{A}'$, if $\alpha \cap \beta \cap P = \emptyset$, then impose $Y[\alpha, \beta] = 0$;
(BZ′ 4) For variables $\alpha_1, \beta_1, \alpha_2, \beta_2 \in \mathcal{A}'$, if $\alpha_1 \cap \beta_1 = \alpha_2 \cap \beta_2$, then impose $Y[\alpha_1, \beta_1] = Y[\alpha_2, \beta_2]$.

– Define

$$\mathrm{BZ}'^k(P) := \left\{ x \in \mathbb{R}^n : \exists Y \in \tilde{\mathrm{BZ}}'^k(P) : \hat{x}(Ye_0) = \hat{x} \right\},$$

and

$$\mathrm{BZ}'^k_+(P) := \left\{ x \in \mathbb{R}^n : \exists Y \in \tilde{\mathrm{BZ}}'^k(P) \cap \mathbb{S}^{\mathcal{A}'}_+ : \hat{x}(Ye_0) = \hat{x} \right\}.$$

Similar to the case of SA^k, BZ'^k can be seen as creating columns that correspond to sets that partition $\mathcal{F}$. While SA^k only generates a partition for each subset of up to k indices, BZ'^k does so for every tier, which is a much broader collection of indices. For a tier S up to size k, it does the same as SA^k and generates $2^{|S|}$ columns corresponding to all possible negations of indices in S. However, for S of size greater than k, it generates a "k-deep" partition of S: a column for $(S \setminus T)|_1 \cap T|_0$ for each $T \subseteq S$ of size up to $k-1$, and the column $S|_{<|S|-k+1}$. Moreover, it also generates columns that partition $(S \setminus T)|_1 \cap T|_0$ for every tier S and every $T \subseteq S$ such that $|T| < k-1$: For each $U \subseteq S$ that is disjoint from T such that $|T| + |U| > k-1$, the algorithm introduces the columns

$$(S \setminus T)|_1 \cap T|_0 \cap (U \setminus U')|_1 \cap U'|_0$$

for all U' of size $\leq (k-1) - |T|$ (so the total number of the negative indices does not exceed $k-1$). It also generates a column for

$$(S \setminus T)|_1 \cap T|_0 \cap U|_{<|U|-(k-1-|T|)}$$

to capture the remainder of the partition.

Notice that in BZ', we have generated exponentially many variables, whereas in the original BZ only polynomially many are selected. The role of walls are also much more important in selecting the variables in BZ, which we have intentionally suppressed in BZ' to make our presentation and analysis easier. We will only use these modified operators to establish negative results, so that the same bounds apply to the original Bienstock–Zuckerberg operators, details of which will be in the full version of this extended abstract.

The main result Bienstock and Zuckerberg achieved with the BZ^k algorithm is when it is applied to set covering problems. Given an inequality $a^T x \geq a_0$ such that $a \geq 0$ and $a_0 > 0$, its *pitch* is defined to be the smallest $j \in \mathbb{N}$ such that

$$S \subseteq \mathrm{supp}(a), |S| \geq j \Rightarrow a^T \chi^S \geq a_0.$$

Also, let $\bar{e}$ denote the all-ones vector of suitable size. Then they showed the following powerful result:

Theorem 2. *Suppose* $P := \{x \in [0,1]^n : Ax \geq \bar{e}\}$ *where* A *is a* $0,1$ *matrix. Then for every* $k \geq 2$*, every valid inequality of* P_I *that has pitch at most* k *is valid for* $\mathrm{BZ}^k(P)$*.*

Note that all inequalities whose coefficients are integral and at most k have pitch no more than k.

2.4 Matching Polytope and the Notion of Rank

We next define the matching polytope of graphs. Given a simple, undirected graph $G = (V, E)$, we define

$$MT(G) := \left\{ x \in [0,1]^E : \sum_{j:\{i,j\} \in E} x_{ij} \leq 1, \ \forall i \in V \right\}.$$

Then the integral points in $MT(G)$ are exactly the incidence vectors of the matchings of G. For any lift-and-project operator Γ, we abbreviate $\Gamma(MT(G))$ as $\Gamma(G)$. Also, for any polytope P, we define the *Γ-rank of P* to be the smallest integer k such that $\Gamma^k(P) = P_I$. The notion of rank gives us a measure of how close P is to P_I with respect to Γ. Moreover, it is useful when comparing the performance of different operators.

3 Some Tools for Upper Bound Analyses

In this section, we present some intermediate results that will help us establish our main results. These tools could be useful in analyzing the lift-and-project ranks of other polytopes as well. Given $j \in \mathbb{Z}_+$, let $[n]_j$ denote all subsets of $[n]$ of size j. Suppose $Y \in \mathbb{S}^{\mathcal{A}'}$ for some $\mathcal{A}' \subseteq \mathcal{A}$. We say that Y is *ℓ-established* if all of the following conditions hold:

(ℓ1) $Y[\mathcal{F}, \mathcal{F}] = 1$;
(ℓ2) $Y \succeq 0$;
(ℓ3) $\mathcal{A}_\ell^+ \subseteq \mathcal{A}'$;
(ℓ4) For any $\alpha, \beta, \alpha', \beta' \in \mathcal{A}_\ell^+$ such that $\alpha \cap \beta = \alpha' \cap \beta', Y[\alpha, \beta] = Y[\alpha', \beta']$.
(ℓ5) For any $\alpha, \beta \in \mathcal{A}_\ell^+, Y[\mathcal{F}, \beta] \geq Y[\alpha, \beta]$.

Notice that any matrix $Y \in \tilde{\mathrm{SA}}_+^{\ell}(P)$ is ℓ-established. A matrix in the lifted space of BZ'_+ is also ℓ-established if all subsets of size up to ℓ are generated as tiers. Given such a matrix, we may define $\mathcal{M} := \bigcup_{i=0}^{2\ell}[n]_i$ and $y \in \mathbb{R}^{\mathcal{M}}$ such that $y_S = Y[S'|_1, S''|_1]$, where S', S'' are subsets of $[n]$ of size at most ℓ such that $S' \cup S'' = S$. Notice that by $(\ell 4)$, the value of y_S does not depend on the choice of S', S''. Finally, we define $Z \in \mathbb{R}^{\{0\}\cup[2\ell]}$ such that

$$Z_i := \sum_{S \subseteq [n]_i} y_S, \ \forall i \geq 0.$$

By $(\ell 1)$, Z_0 is always equal to 1. Also note that, $Z_1 = \sum_{i=1}^{n} Y[i|_1, \mathcal{F}]$. We see that the entries of Z are related to each other. For example, if $\hat{x}(Ye_{\mathcal{F}})$ is an integral $0,1$ vector, then by $(\ell 5)$ we know that $y_S \leq 1$ for all S, and $y_S > 0$ only if $y_{\{i\}} = 1, \ \forall i \in S$. Thus, we can infer that

$$Z_j = \sum_{S \in [n]_j} y_S \leq \binom{Z_1}{j}, \ \forall j \in [2\ell].$$

Next, we show that the positive semidefiniteness of Y also forces the Z_i's to relate to each other, somewhat similarly to the above. The following result would be more intuitive by noting that $\binom{p}{i+1}/\binom{p}{i} = \frac{p-i}{i+1}$.

Proposition 3. *Suppose $Y \in \mathbb{S}^{\mathcal{A}'}$ is ℓ-established, and y, Z are defined as above. If there exists $p \in \mathbb{R}_+$ such that*

$$Z_{i+1} \leq \left(\frac{p-i}{i+1}\right) Z_i, \ \forall i \in [\ell, 2\ell - 1],$$

then $Z_i \leq \binom{p}{i}, \ \forall i \leq 2\ell$. In particular, $Z_1 \leq p$.

An immediate but noteworthy implication of Proposition 3 is the following:

Corollary 4. *Suppose $Y \in \mathbb{S}^{\mathcal{A}'}$ is ℓ-established, and y, Z are defined as above. If there exists $p \in [0, \ell]$ such that $Z_i = 0, \ \forall i > p$, then $Z_1 \leq p$.*

4 The SA_+-rank, the Las-rank, and the Theta-rank of the Matching Polytope

We now turn to our main results and determine the lift-and-project ranks of $MT(K_{2n+1})$ for various operators. First, we study the SA_+-rank.

Theorem 5. *The SA_+-rank of $MT(K_{2n+1})$ is at least $\left\lfloor \frac{n}{2} \right\rfloor + 1$.*

Proof (sketch). We prove our claim by showing that $\frac{1}{4n}\bar{e} \in \mathrm{SA}_+^n(K_{4n+1})$, implying that $MT(K_{4n+1})$ has SA_+-rank at least $n+1$, from which our

assertion follows. Define $Y \in \mathbb{S}^{\mathcal{A}_n}$ such that $Y[\emptyset,\emptyset] := 1$, and $Y[S_1|_1, S_2|_1] := \prod_{i=1}^{|S_1 \cup S_2|} \frac{1}{4n+2-2i}$ if $S_1 \cup S_2$ is a matching and 0 otherwise. Also, set $Y[S_1|_1 \cap T_1|_0, S_2|_1 \cap T_2|_0] := \sum_{U \subseteq T_1 \cup T_2} (-1)^{|U|} Y[S_1 \cup (U \cap T_1)|_1, S_2 \cup (U \cap T_2)|_1]$.

Notice that $\bar{x}(Ye_{\mathcal{F}}) = \frac{1}{4n}\bar{e}$, and (SA 1), (SA 3), (SA$_+$ 2) and (SA$_+$ 3) all hold by the construction of Y. Also, it was shown in [MS09] that (SA 2) holds. Thus, it only remains to verify that $Y \succeq 0$. By exploiting the linear dependencies of the columns of Y and the symmetries of the complete graph, the task of showing $Y \succeq 0$ can be reduced to showing $Y' \succeq 0$, where

$$Y'[i,j] := \sum_{\substack{S_1,S_2 \subseteq E,\\ |S_1|=i, |S_2|=j}} Y[S_1|_1, S_2|_1], \ \forall i,j \in \{0,1,\ldots,n\}.$$

It can be checked that $Y'[i,j] = \binom{4n+1}{i}\binom{4n+1}{j}$ for all integers $i,j \in [0,n]$. Hence $Y' = (Y'e_0)(Y'e_0)^T$ and our claim follows. □

Next, we employ the upper bound proving techniques from Section 3 and the notion of ℓ-established to prove the next result.

Proposition 6. *The* SA$_+$*-rank of* $MT(K_{2n+1})$ *is at most* $n - \left\lfloor \frac{\sqrt{2n+1}-1}{2} \right\rfloor$.

Somewhat surprisingly, a lower bound of the Las-rank of the matching polytope follows almost immediately from the proof of Theorem 5.

Theorem 7. *The* Las*-rank and Theta-rank of* $MT(K_{2n+1})$ *is at least* $\lfloor \frac{n}{2} \rfloor$ *and at most* n.

5 The BZ$_+$-rank of the Matching Polytope

Next, we turn to the BZ$_+$-rank of the matching polytope. Before we do that, it is beneficial to characterize some of the variables generated by the stronger BZ$'^k_+$ that obviously do not help generate any cuts. We say that a tier S generated by BZ$'^k$ is *ℓ-useless* if

1. For all $T \subseteq S$ such that $|T| \leq k-1$, $(S \setminus T)|_1 \cap P = \emptyset$;
2. $\sum_{i \in S} x_i \leq |S| - k$ is valid for SA$^\ell(P_{\mathcal{O}})$.

Then, we have the following:

Lemma 8. *Suppose there exists* $\ell \in \mathbb{Z}_+$ *such that all tiers* S *generated by* BZ$'^k$ *of size greater than* ℓ *are* ℓ*-useless. Then*

$$\text{BZ}'^k(P) \supseteq \text{SA}^{2\ell}(P_{\mathcal{O}}) \text{ and } \text{BZ}'^k_+(P) \supseteq \text{SA}^{\ell}_+(P_{\mathcal{O}}).$$

We are now ready to approximate the BZ$_+$-rank of $MT(K_{2n+1})$, to within a constant factor.

Theorem 9. *Suppose* $G = K_{2n+1}$. *Then the* BZ$_+$*-rank of* $MT(G)$ *is between* $\sqrt{n}$ *and* $\sqrt{2n}+1$.

In fact, we prove that the above lower bound applies to the stronger BZ′. We also remark that, in general, adding redundant inequalities to the system $Ax \leq b$ would generate more obstructions and walls, and thus could improve the performance of BZ_+. In fact, if we let $G := K_{2n+1}$ and include every valid inequality of $MT(G)$ in the initial description of the polytope $MT(G)$, then any matrix $Y \in \tilde{\mathrm{BZ}}_+^2(G)$ is actually n-established, which implies that $\hat{x}(Ye_{\mathcal{F}}) \in K(MT(G)_I)$.

6 The BZ-rank of the Stable Set Polytope

Another family of polytopes related to graphs that has been studied extensively is the family of stable set polytopes. Given a graph $G = (V, E)$, its *fractional stable set polytope* is defined to be

$$FRAC(G) := \left\{x \in [0,1]^V \ : \ x_i + x_j \leq 1, \ \forall \{i,j\} \in E\right\}.$$

Then the *stable set polytope* $STAB(G) := FRAC(G)_I$ is precisely the convex hull of incidence vectors of stable sets of G. For the complete graph $G := K_n$, $FRAC(G)$ is known to have rank 1 with respect to the LS_+ and Las operators. Proposition 1 implies that it also has SA_+-rank 1. Its BZ_+-rank is 1 as well, as it is not hard to see that SA_+^1 is refined by BZ_+^1. However, the rank is known to be $\Theta(n)$ for all other operators that yield only polyhedral relaxations, such as SA and Lovász–Schrijver's N operator [LS91]. We show that BZ operator also has the same property.

Theorem 10. *Suppose G is the complete graph on $n \geq 5$ vertices. Then the* BZ*-rank of $FRAC(G)$ is either $\left\lceil \frac{n}{2} - 1 \right\rceil$ or $\left\lceil \frac{n}{2} \right\rceil$.*

Again, the lower bound of the above also applies to BZ′.

References

[ABN04] Aguilera, N.E., Bianchi, S.M., Nasini, G.L.: Lift and project relaxations for the matching and related polytopes. Discrete Appl. Math. 134(1-3), 193–212 (2004)

[Bal98] Balas, E.: Disjunctive programming: properties of the convex hull of feasible points. Discrete Appl. Math. 89(1-3), 3–44 (1998)

[BCC93] Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. Math. Programming 58(3, Ser. A), 295–324 (1993)

[BGM10] Benabbas, S., Georgiou, K., Magen, A.: The Sherali-Adams system applied to vertex cover: why Borsuk graphs fool strong LPs and some tight integrality gaps for SDPs (2010) (extended Abstract)

[BZ04] Bienstock, D., Zuckerberg, M.: Subset algebra lift operators for 0-1 integer programming. SIAM J. Optim. 15(1), 63–95 (2004)

[CD01] Cook, W., Dash, S.: On the matrix-cut rank of polyhedra. Math. Oper. Res. 26(1), 19–30 (2001)

[Che05] Cheung, K.K.H.: On Lovász-Schrijver lift-and-project procedures on the Dantzig-Fulkerson-Johnson relaxation of the TSP. SIAM J. Optim. 16(2), 380–399 (2005)

[Che07] Cheung, K.K.H.: Computation of the Lasserre ranks of some polytopes. Math. Oper. Res. 32(1), 88–94 (2007)

[GMPT06] Georgiou, K., Magen, A., Pitassi, T., Tourlakis, I.: Tight integrality gaps for vertex cover SDPs in the Lovász-Schrijver hierarchy. Electronic Colloquium on Computational Complexity (ECCC) 13(152) (2006)

[GPT10] Gouveia, J., Parrilo, P.A., Thomas, R.R.: Theta bodies for polynomial ideals. SIAM Journal on Optimization 20(4), 2097–2118 (2010)

[GT01] Goemans, M.X., Tunçel, L.: When does the positive semidefiniteness constraint help in lifting procedures? Math. Oper. Res. 26(4), 796–815 (2001)

[HT08] Hong, S.-P., Tunçel, L.: Unification of lower-bound analyses of the lift-and-project rank of combinatorial optimization polyhedra. Discrete Appl. Math. 156(1), 25–41 (2008)

[Las01] Lasserre, J.B.: An explicit exact SDP relaxation for nonlinear 0-1 programs. In: Aardal, K., Gerards, B. (eds.) IPCO 2001. LNCS, vol. 2081, pp. 293–303. Springer, Heidelberg (2001)

[Lau02] Laurent, M.: Tighter linear and semidefinite relaxations for max-cut based on the Lovász-Schrijver lift-and-project procedure. SIAM J. Optim. 12(2), 345–375 (2001/2002)

[Lau03] Laurent, M.: A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming. Math. Oper. Res. 28(3), 470–496 (2003)

[LS91] Lovász, L., Schrijver, A.: Cones of matrices and set-functions and 0-1 optimization. SIAM J. Optim. 1(2), 166–190 (1991)

[LT03] Lipták, L., Tunçel, L.: The stable set problem and the lift-and-project ranks of graphs. Math. Program. 98(1-3, Ser. B), 319–353 (2003); Integer programming (Pittsburgh, PA, 2002)

[MS09] Mathieu, C., Sinclair, A.: Sherali-Adams relaxations of the matching polytope. In: STOC 2009. ACM Press, New York (2009)

[SA90] Sherali, H.D., Adams, W.P.: A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J. Discrete Math. 3(3), 411–430 (1990)

[ST99] Stephen, T., Tunçel, L.: On a representation of the matching polytope via semidefinite liftings. Math. Oper. Res. 24(1), 1–7 (1999)

[STT06] Schoenebeck, G., Trevisan, L., Tulsiani, M.: A linear round lower bound for Lovász-Schrijver SDP relaxations of vertex cover. In: IEEE Conference on Computational Complexity, pp. 6–98. IEEE Computer Society, Los Alamitos (2006)

[Zuc03] Zuckerberg, M.: A Set Theoretic Approach to Lifting Procedures for 0,1 Integer Programming. PhD thesis, Columbia University (2003)

A Probabilistic Analysis of the Strength of the Split and Triangle Closures

Amitabh Basu[1], Gérard Cornuéjols[2,⋆], and Marco Molinaro[2,⋆⋆]

[1] University of California, Davis
[2] Carnegie Mellon University

Abstract. In this paper we consider the relaxation of the corner polyhedron introduced by Andersen et al., which we denote by RCP. We study the relative strength of the split and triangle cuts of RCP's. Basu et al. showed examples where the split closure can be arbitrarily worse than the triangle closure under a 'worst-cost' type of measure. However, despite experiments carried out by several authors, the usefulness of triangle cuts in practice has fallen short of its theoretical strength.

In order to understand this issue, we consider two types of measures between the closures: the 'worst-cost' one mentioned above, where we look at the weakest direction of the split closure, and the 'average-cost' measure which takes an average over all directions. Moreover, we consider a natural model for generating random RCP's. Our first result is that, under the worst-cost measure, a random RCP has a weak split closure with reasonable probability. This shows that the bad examples given by Basu et al. are not pathological cases. However, when we consider the average-cost measure, with high probability both split and triangle closures obtain a very good approximation of the RCP. The above result holds even if we replace split cuts by the simple split or Gomory cuts. This gives an indication that split/Gomory cuts are indeed as useful as triangle cuts.

1 Introduction

Consider an IP in standard form:

$$\begin{aligned} \min\, & ay \\ Ay &= b \\ y \geq 0,\quad & y \in \mathbb{Z}^d. \end{aligned} \tag{IP}$$

Suppose that B is an optimal basis for the LP relaxation of (IP). Rewriting IP in tableaux form with respect to B (i.e., pre-multiplying the system by B^{-1}) we

⋆ Supported by NSF grant CMMI1024554 and ONR grant N00014-09-1-0033.
⋆⋆ Supported by NSF grant CMMI1024554 and a Mellon Fellowship.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 27–38, 2011.

obtain the equivalent system

$$\begin{aligned} &\min \bar{a}_N y_N \\ &y_B = \bar{b} - \bar{N} y_N \\ &y \geq 0, \quad y \in \mathbb{Z}^d \end{aligned} \tag{IP'}$$

where $\bar{a}_N \geq 0$ due to the optimality of B.

In [1], Andersen et al. introduced a relaxation of (IP') which we call Relaxed Corner Polyhedron (RCP). This is a further weakening of the Corner relaxation [12] where: (i) the integrality constraints are dropped for the nonbasic variables and (ii) the non-negativity constraints are dropped for the basic variables. Rewriting in a different way, an RCP is a MIP of the form

$$\begin{aligned} &\min cs \\ &x = f + \sum_{j=1}^{n} r^j s_j \\ &x \in \mathbb{Z}^m, \quad s \geq 0 \end{aligned} \tag{RCP}$$

with $c \geq 0$.

An RCP is defined by the vectors $f, r^1, \ldots, r^n$ and the cost vector c. We call a tuple $\langle f, r^1, r^2, \ldots, r^n \rangle$ an *ensemble*. Given an ensemble $\mathcal{E}$ and cost vector c, we use $RCP(\mathcal{E}, c)$ to denote the corresponding RCP.

In this work, we are interested in comparing the cost of an optimal solution to $RCP(\mathcal{E}, c)$ against the cost of an optimal solution to some of its relaxations. In order to simplify things, we work over the projection onto the s-space; the crucial property given by the structure of RCP's is that the projection of any solution onto the s-space has the same cost as the original solution. We define $P(\mathcal{E})$ as the projection of the convex hull of feasible solutions of $RCP(\mathcal{E}, c)$ onto the s-space. In other words, $P(\mathcal{E}) = \text{conv}(\{s \geq 0 \,|\, f + \sum_{j=1}^{n} r^j s_j \in \mathbb{Z}^m\})$.

Intersection cuts. Let X be a closed, full-dimensional convex set in $\mathbb{R}^m$ which: (i) contains f in its interior and (ii) does not contain any integer point in its interior. The functional $\psi_X : \mathbb{R}^m \to \mathbb{R}$ is defined as

$$\psi_X(r) = \inf\{\lambda > 0 : f + \frac{r}{\lambda} \in X\}. \tag{1}$$

The inequality $\sum_{j=1}^{n} \psi_X(r^j) s_j \geq 1$ is valid for $P(\mathcal{E})$, since it is an intersection cut [2]. Moreover, it was shown in [8] that all minimal inequalities of $P(\mathcal{E})$ are of this form (for some X). Since these inequalities are constructed based on convex sets in $\mathbb{R}^m$, this gives a geometric way of analyzing $P(\mathcal{E})$.

One important family of inequalities is derived from sets X which are 'splits', that is, X is of the form $\{x : b_1 \leq ax \leq b_2\}$. We call them split cuts. The split closure of $P(\mathcal{E})$, denoted by $S(\mathcal{E})$, is the intersection of all split cuts. We also consider cuts from *simple* splits, that is, splits of the form $\{x : \lfloor b \rfloor \leq e^i x \leq \lceil b \rceil\}$ where $b \in \mathbb{R} \setminus \mathbb{Z}$ and e^i is the ith canonical vector in $\mathbb{R}^m$. We denote the closure of these cuts by $G(\mathcal{E})$. Finally, for the case $m = 2$, another important family is when X is a triangle. The triangle closure is denoted by $T(\mathcal{E})$.

Strength of relaxations. In [6] the authors showed a family of RCP's whose split closure gets arbitrarily weaker than the triangle closure. In their work, a 'worst-case' type of measure is used to compare these two closures. Interestingly, split cuts for (IP) derived by taking into account all integrality constraints actually perform very well in practice [4]. The apparent strength of triangle cuts suggests that even stronger cuts for (IP) can be obtained from them.

Motivated by this observation, several authors recently started to test experimentally the effectiveness of the triangle cuts [11,3,5,10]. Despite some positive results obtained by Balas and Qualizza in [3], the usefulness of the triangle cuts has been very far from those indicated by theory.

However, it is still not clear if the disappointing performance of triangle cuts is due to insufficient separation heuristics or it is really an innate property of these cuts, although the latter seems more likely. Our goal in this paper is to further explore the relative strength of the split and triangle closure from a theoretical perspective.

Our results. Informally, the results presented in this paper can be described as follows. We consider two ways of comparing the closures $P(\mathcal{E})$, $G(\mathcal{E})$, $S(\mathcal{E})$ and $T(\mathcal{E})$. Roughly, in the 'worst-cost' measure we look at the direction where the relaxation is weakest, while in the 'average-cost' measure we take an average over all directions. *We show that the strength of the split closure is very dependent on which measure is used to perform the comparison.*

First we consider the worst-cost measure and $m = 2$. We show that with reasonable probability the split closure of a random RCP is very weak when compared to the triangle closure. This generalizes the bad examples from [6] and shows that they are not as pathological as one could think. On the other hand, with respect to the average-cost measure we show that, with high probability, the simple split closure $G(\mathcal{E})$ is a very good approximation of $P(\mathcal{E})$. Since $P(\mathcal{E}) \subseteq T(\mathcal{E}) \subseteq S(\mathcal{E}) \subseteq G(\mathcal{E})$, this shows in particular that the split and the triangle closure are very similar under the average-cost measure and also approximate the integer hull very well. This gives a partial justification why triangle cuts seem to have a similar performance to split cuts in practice.

We remark that, due to space constraints, the proof of some claims are omitted; they can be found in [7].

Related work. Two recent papers address the fundamental question of comparing the strengths of triangle and split cuts from a probabilistic point of view.

He et al. [13] use the same random model for generating RCP's, but a different measure to compare the strength of cuts, comparing the random coefficients of the inequalities induced by the randomness of the rays. Their analysis does not consider the important triangles of Type 3. Although the results cannot be directly compared, their paper also indicates that split cuts perform at least as well as some classes of triangles.

Del Pia et al. [14] base their analysis on the lattice width of the underlying convex set. They show that the importance of triangle cuts generated from

Type 2 triangles (the same family which was considered in [6]) decreases with decreasing lattice width, on average. They also have results for triangles of Type 3 and for quadrilaterals.

Our approach is very different from these two papers.

2 Preliminaries

Measures of strength. Let A and B be convex relaxations of $P(\mathcal{E})$ such that $A, B \subseteq \mathbb{R}^n_+$. A closed, convex set $X \subseteq \mathbb{R}^n_+$ is said to be of blocking type if $y \geq x \in X$ implies $y \in X$. It is well-known that the recession cone of $P(\mathcal{E})$ is $\mathbb{R}^n_+$(see [9]) and hence $P(\mathcal{E})$, A and B are convex sets of blocking type. A traditional measure of strength for integer programs is the *integrality gap*, which compares the ratio of the minimization over the IP and its linear relaxation. More generally, we define the *gap* between A and B with respect to the cost vector c as:

$$\text{gap}(A, B, c) = \frac{\inf\{cs : s \in A\}}{\inf\{cs : s \in B\}}. \tag{2}$$

Notice that this value is greater than 1 if A is stronger than B, i.e. $A \subseteq B$. We define the gap to be $+\infty$ if A is empty or $\inf\{cs : s \in B\} = 0$.

Based on this idea, we can define the *worst-cost* measure between the two relaxations A and B as the worst possible gap over all non-negative cost vectors:

$$\text{wc}(A, B) = \sup_{c \in \mathbb{R}^m_+} \{\text{gap}(A, B, c)\} \tag{3}$$

$$= \sup_{c \in [0,1]^m} \{\text{gap}(A, B, c)\}, \tag{4}$$

where the second equation follows from the fact that the ratios are preserved under positive scaling of the cost vectors. Note that for convex sets of blocking type, only non-negative cost vectors have bounded optimum, hence we will restrict ourselves to this case.

Now we define another (more robust) measure of strength which tries to capture the average strength with respect to different costs. Consider a distribution $\mathcal{C}$ over vectors in $\mathbb{R}^m_+$ and let $c \sim \mathcal{C}$ indicate that c is a random vector with distribution $\mathcal{C}$. Then, the *average-cost* measure between A and B is defined by

$$\text{avg}(A, B, \mathcal{C}) = \mathbb{E}_{c \sim \mathcal{C}} [\text{gap}(A, B, c)]. \tag{5}$$

In the sequel we study the worst-cost and average-cost strength of the split and triangle closures for random RCP's. We define our model for random RCP's next.

Random model. Let $\mathcal{D}^m_n$ denote the distribution of ensembles $\langle f, r^1, \ldots, r^n \rangle$ where f is picked uniformly from $[0, 1]^m$ and each of $r^1, \ldots, r^n$ is picked independently and uniformly at random from the set of rational unit vectors in $\mathbb{R}^m$. We make a note here that the rays in RCP can be assumed to be unit vectors, by suitably scaling the cost coefficients. In other words, given an ensemble

$\mathcal{E}$ and a cost vectors c, there exists an ensemble $\mathcal{E}'$ all of whose rays are unit vectors and a cost vector c', such that the optimal value of $\mathrm{RCP}(\mathcal{E}, c)$ equals to the optimal value of $\mathrm{RCP}(\mathcal{E}', c')$. Moreover, there exists an invertible affine transformation $A : \mathbb{R}^n \to \mathbb{R}^n$ such that $P(\mathcal{E}) = A(P(\mathcal{E}'))$. Hence, in our model we assume that the rays are sampled from the set of rational unit vectors. When the dimension is 2 we write $\mathcal{D}_n$ for the distribution, omitting the superscript.

3 Worst-Cost Measure in $\mathbb{R}^2$

The main result of this section is that, for a significant fraction of the RCP's in the plane, $S(\mathcal{E})$ is significantly worse than $T(\mathcal{E})$ based on the worst-cost measure.

Theorem 1. *For any $\alpha \geq 1$ and $\beta \in [0, 1]$, a random ensemble $\mathcal{E} \sim \mathcal{D}_n$ satisfies*

$$\Pr\left(\mathrm{wc}(T(\mathcal{E}), S(\mathcal{E})) \geq \alpha\right) \geq \left[1 - 2\left(1 - g(\frac{\beta}{4\alpha})\right)^n\right]\left[\frac{1-\beta}{\alpha} - \frac{1-\beta^2}{4\alpha^2}\right],$$

where

$$g(x) = \frac{1}{2\pi}\left(\frac{x}{0.75 - (2-\sqrt{2})x} - \frac{x}{1-(2-\sqrt{2})x}\right).$$

Notice that this bound increases as n grows. In the limit $n \to \infty$, and using the optimal choice $\beta \to 0$, the bound becomes $1/\alpha - 1/4\alpha^2$. To obtain an idea about the probabilities in the above theorem, Table 1 presents the bound obtained for different values of n and α.

Remark 1. Since $P(\mathcal{E}) \subseteq T(\mathcal{E})$, Theorem 1 implies that the same statement holds when $T(\mathcal{E})$ is replaced by $P(\mathcal{E})$, showing that the split closure is a bad approximation of the integer hull with the same probability.

Table 1. Values of the bound of Theorem 1 for different values of n and approximation factor α. The value of β in every entry was chosen empirically and attempts to optimize the bound.

n	α	β	Pr
100	1.5	0.37	25.7%
100	2	0.43	16.7%
500	2	0.16	33.6 %
500	3	0.22	21.3%
1000	2	0.01	37.7%
1000	3	0.14	25.0 %
1000	4	0.17	18.2 %
$+\infty$	2	0	43.75 %
$+\infty$	4	0	30.56 %

The way to prove this result is to consider a particular (deterministic) ensemble $\langle f, r^1, r^2\rangle$ which is 'bad' for the split closure and show that it appears with significant probability in a random ensemble. We employ the following monotonicity property to transfer the 'badness' to the whole RCP.

Lemma 1. *Consider an ensemble* $\mathcal{E} = \langle f, r^1, \ldots, r^n \rangle$ *and let* $\mathcal{E}' = \langle f, r^{i_1}, r^{i_2}, \ldots, r^{i_k} \rangle$ *be a subensemble of it. Then* $\text{wc}(T(\mathcal{E}), S(\mathcal{E})) \geq \text{wc}(T(\mathcal{E}'), S(\mathcal{E}'))$.

3.1 A Bad Ensemble for the Split Closure

First, we introduce the following notation: Given a point f and a ray r, we say that $f + r$ *crosses* a region $R \subseteq \mathbb{R}^n$ if there is $\lambda \geq 0$ such that $f + \lambda r \in R$.

In this part we will focus on ensembles $\mathcal{E} = \langle f, r^1, r^2 \rangle$ where $f \in (0,1)^2$, and $f + r^1$ and $f + r^2$ cross the open segment connecting $(0,0)$ to $(0,1)$. The high-level idea is the following. Suppose that r^1 an r^2 have x_1-value equal to -1 and consider a lattice-free triangle T containing the points $f + r^1$ and $f + r^2$, and also containing f in its interior. This triangle gives an inequality which is at least as strong as $s_1 + s_2 \geq 1$, hence we have a lower bound of 1 for minimizing $s_1 + s_2$ over the triangle closure $T(\mathcal{E})$. However, further assume that the angle between rays r^1 and r^2 is large. Then we can see that any split that contains f in its interior will have a very large coefficient for either s_1 or s_2. More specifically, suppose that there is a large M such that, for every inequality $\psi(r^1)s_1 + \psi(r^2)s_2 \geq 1$ coming from a split, we have $\max\{\psi(r^1), \psi(r^2)\} \geq M$. Then the point $(s_1, s_2) = (1/M, 1/M)$ satisfies every such inequality and hence is feasible for the split closure $S(\mathcal{E})$; this gives an upper bound of $2/M$ for minimizing $s_1 + s_2$ over the split closure. Then using the choice of $c = [1, 1]$ in the maximization in (3) gives $\text{wc}(T(\mathcal{E}), S(\mathcal{E})) \geq M/2$. The following lemma formalizes the observation that if r^1 and r^2 are spread out then the split closure is weak.

Lemma 2. *Consider an ensemble* $\mathcal{E} = \langle f, r^1, r^2 \rangle$ *where* $f = (f_1, f_2) \in (0,1)^2$, $r^1 = c_1(-1, t_1)$ *and* $r^2 = c_2(-1, t_2)$ *with* $c_1, c_2 \geq 0$ *and* $t_1 \geq t_2$. *Moreover, assume that both* $f + r^1$ *and* $f + r^2$ *cross the left facet of the unit square. Then*

$$\min\{c_1 s_1 + c_2 s_2 : (s_1, s_2) \in S(\mathcal{E})\} \leq \frac{2}{t_1 - t_2}.$$

Now we are ready to establish the main lemma of this section, which exhibits bad ensembles for the split closure.

Lemma 3. *Consider an ensemble* $\mathcal{E} = \langle f, r^1, r^2 \rangle$ *where* $f = (f_1, f_2) \in (0,1)^2$. *Suppose that* $f + r^1$ *crosses the open segment connecting* $(0, 1 - \epsilon)$ *and* $(0, 1)$ *and that* $f + r^2$ *crosses the open segment connecting* $(0, 0)$ *and* $(0, \epsilon)$, *for some* $0 < \epsilon < 1$. *Then* $\text{wc}(T(\mathcal{E}), S(\mathcal{E})) \geq (1 - 2\epsilon)/2f_1$.

Proof. Let $v^1 = (-1, t_1)$, $v^2 = (-1, t_2)$ and let $c_1, c_2 \geq 0$ be such that $r^1 = c_1 v^1$ and $r^2 = c_2 v^2$. By the assumptions on the rays, we have $t_1 \geq t_2$.

Consider the rays $\underline{v}^1 = (-1, \underline{t}_1)$ and $\underline{v}^2 = (-1, \underline{t}_2)$ such that $f + \underline{v}^1$ crosses $(0, 1 - \epsilon)$ and $f + \underline{v}^2$ crosses $(0, \epsilon)$.

Notice that $t_1 \geq \underline{t}_1 \geq \underline{t}_2 \geq t_2$, implying that $t_1 - t_2 \geq \underline{t}_1 - \underline{t}_2$. Moreover, using similarity of triangles we obtain that $\underline{t}_1 - \underline{t}_2 = \frac{1-2\epsilon}{f_1}$. Therefore, $t_1 - t_2 \geq (1 - 2\epsilon)/f_1$.

Employing Lemma 2 over $\mathcal{E}'$ gives $\min\{c_1 s_1 + c_2 s_2 : (s_1, s_2) \in S(\mathcal{E}')\} \leq 2f_1/(1-2\epsilon)$. In contrast, $\min\{c_1 s_1 + c_2 s_2 : (s_1, s_2) \in T(\mathcal{E}')\} \geq 1$, because of the inequality $c_1 s_1 + c_2 s_2 \geq 1$ derived from the lattice-free triangle with vertices $f+v^1$, $f+v^2$ and $f-(\gamma, 0)$ for some $\gamma > 0$. Notice that such γ exists because $f+v^1$ and $f+v^2$ do not cross the points $(0,1)$ and $(0,0)$ respectively. Using the cost vector $c = [c_1, c_2]$, we obtain the desired bound $\text{wc}(T(\mathcal{E}), S(\mathcal{E})) \geq (1-2\epsilon)/2f_1$.

3.2 Probability of Bad Ensembles

Using the ensemble constructed in the previous section and the monotonicity property from Lemma 1, we now analyze the probability that a random ensemble $\mathcal{E} \sim \mathcal{D}_n$ is bad for the split closure. Let Δ denote the triangle in $\mathbb{R}^2$ with vertices $(0,0)$, $(0,1)$, $(1/2, 1/2)$.

Lemma 4. *Let $\mathcal{E} = \langle f, r^1, \ldots, r^n \rangle$ be a random ensemble from $\mathcal{D}_n$, where $f = (f_1, f_2)$. Then for all $\bar{f} = (\bar{f}_1, \bar{f}_2) \in \Delta$ and all $\epsilon \in (0, 1/2)$, we have*

$$\Pr\left(\text{wc}(T(\mathcal{E}), S(\mathcal{E})) \geq \frac{1-2\epsilon}{\bar{f}_1} \middle| f = \bar{f}\right) \geq 1 - 2\left(1 - g(\bar{f}_1)\right)^n,$$

where

$$g(x) = \frac{1}{2\pi}\left(\frac{x}{1-\epsilon-(2-\sqrt{2})x} - \frac{x}{1-(2-\sqrt{2})x}\right).$$

Proof. Let us call *portals* the open segment connecting $(0, 1-\epsilon)$ and $(0,1)$ and the open segment connecting $(0,\epsilon)$ and $(0,0)$. Due to Lemmas 1 and 3 it suffices to lower bound the probability that a random ensemble has rays r^i and r^j such that $f + r^i$ crosses one portal and $f + r^j$ crosses the other portal.

Consider a ray r^i; the probability that $f + r^i$ crosses the open segment connecting $(0, 1-\epsilon)$ and $(0,1)$ equals to $\theta/2\pi$, where θ is the angle between the vectors $(0, 1-\epsilon) - \bar{f}$ and $(0,1) - \bar{f}$. Expressing θ using the function arctan(.) and then using the concavity of the later, one can prove that $\theta \geq 2\pi g(\bar{f}_1)$ (we refer to the full version of the paper for a proof). Therefore, the probability that $\bar{f} + r^i$ crosses the open segment connecting $(0, 1-\epsilon)$ and $(0,1)$ is at least $g(\bar{f}_1)$. By symmetry, we can also prove that the probability that $\bar{f} + r^i$ crosses the open segment connecting $(0,\epsilon)$ and $(0,0)$ is also at least $g(\bar{f}_1)$; this bounds also holds for this case because it is independent of $\bar{f}_2$.

Let B_1 denote the event that no ray of $\mathcal{E}$ crosses the open segment connecting $(0, 1-\epsilon)$ and $(0,1)$ and let B_2 denote the even that no ray of $\mathcal{E}$ crosses the open segment connecting $(0,\epsilon)$ and $(0,0)$. Using our previous bound we obtain that $\Pr(B_1) \leq (1 - g(\bar{f}_1))^n$, and the same lower bound holds for $\Pr(B_2)$. Notice that the probability that $\mathcal{E}$ has rays r^i and r^j such that $f + r^i$ and $f + r^j$ cross distinct portals is $1 - \Pr(B_1 \vee B_2)$; from union bound we get that this probability is at least $1 - 2(1 - g(\bar{f}_1))^n$. This concludes the proof of the lemma.

3.3 Proof of Theorem 1

In order to conclude the proof of Theorem 1 we need to remove the conditioning in the previous lemma. To make progress towards this goal, for $t \in [0, 1/2]$ let $\Delta_t = \Delta \cap \{(x_1, x_2) : x_1 \leq t\}$. It is easy to see that the area of Δ_t equals $(1-t)t$. Now it is useful to focus on the set $\Delta_t \setminus \Delta_{\beta t}$, for some $\beta \in [0, 1]$, since we can bound the probability that a uniform point lies in it and Lemma 4 is still meaningful. Using the independence properties of the distribution $\mathcal{D}_n$ we get that for every $\beta \in [0, 1]$ and $\epsilon \in (0, 1/2)$ a random ensemble $\mathcal{E} = \langle f, r^1, \ldots, r^n \rangle \sim \mathcal{D}_n$ satisfies:

$$\begin{aligned}
&\Pr\left(\mathrm{wc}(T(\mathcal{E}), S(\mathcal{E})) \geq \frac{1-2\epsilon}{2t} \middle| f \in \Delta\right) \\
&\geq \Pr\left(\mathrm{wc}(T(\mathcal{E}), S(\mathcal{E})) \geq \frac{1-2\epsilon}{2t} \middle| f \in \Delta_t \setminus \Delta_{\beta t}\right) \Pr\left(f \in \Delta_t \setminus \Delta_{\beta t} \middle| f \in \Delta\right) \\
&\geq \left[1 - 2\left(1 - g\left(\beta t\right)\right)^n\right] \cdot 4 \cdot \left[(1-t)t - (1-\beta t)\beta t\right],
\end{aligned}$$

where the first inequality follows from the fact that $\Delta_t \setminus \Delta_{\beta t} \subseteq \Delta$ and the second inequality follows from the fact that $\beta t \leq f_1 \leq t$ and that the function $g(x)$ is increasing in x.

Finally, notice that this bound holds for all four 90-degree rotations of Δ around the point $(1/2, 1/2)$; this is because of the symmetries of $\mathcal{D}_n$. Thus, by law of total probability we can remove the last conditioning. Using $\epsilon = 1/4$ and $\alpha = 1/4t$ we then obtain Theorem 1. We remark that we fixed the value of ϵ in order to simplify the expression in the theorem and that the value $1/4$ was chosen experimentally in order to obtain good bounds specially for reasonably small values of n.

Since $T(\mathcal{E})$ is a relaxation of $P(\mathcal{E})$, as a corollary of the theorem we obtain a bound on the probability that the split closure is bad for random RCP's.

Corollary 1. *For any $\alpha \geq 1$ and $\beta \in [0, 1]$, a random ensemble $\mathcal{E} \sim \mathcal{D}_n$ satisfies*

$$\Pr\left(\mathrm{wc}(P(\mathcal{E}), S(\mathcal{E})) \geq \alpha\right) \geq \left[1 - 2\left(1 - g(\frac{\beta}{4\alpha})\right)^n\right] \left[\frac{1-\beta}{\alpha} - \frac{1-\beta^2}{4\alpha^2}\right],$$

4 Average-Cost Measure

For $\epsilon > 0$ we define the product distribution $\mathcal{P}_\epsilon$ over $[\epsilon, 1]^n$ where a vector is obtained by taking each of its n coefficients independently uniformly in $[\epsilon, 1]$. In this section we show that $avg(P(\mathcal{E}), G(\mathcal{E}), \mathcal{P}_\epsilon)$ is small for most ensembles $\mathcal{E}$ in $\mathcal{D}_n^m$.

Theorem 2. *Fix reals $\epsilon > 0$ and $\alpha > 1$ and an integer $m > 0$. Then for large enough n,*

$$\Pr_{\mathcal{E} \sim \mathcal{D}_n^m}\left(\mathrm{avg}(P(\mathcal{E}), G(\mathcal{E}), \mathcal{P}_\epsilon) \leq \alpha\right) \geq 1 - \frac{1}{n}.$$

Remark 2. Since $P(\mathcal{E}) \subseteq T(\mathcal{E}) \subseteq S(\mathcal{E}) \subseteq G(\mathcal{E})$, Theorem 2 implies that $S(\mathcal{E})$ is a good approximation of $T(\mathcal{E})$ and $P(\mathcal{E})$ under the average-cost measure. In fact, the same statement as Theorem 2 holds for any pair of bodies from $P(\mathcal{E}), T(\mathcal{E}), S(\mathcal{E}), G(\mathcal{E})$ with the appropriate containments. We remark that the property that the cost vector is bounded away from zero in every coordinate is crucial in our analysis. This is needed because the ratio in (2) can become ill-defined in the presence of rays of zero cost.

The high level idea for proving the theorem is the following. Consider an ensemble $\mathcal{E} = \langle f, r^1, \ldots, r^n \rangle$. Define $\hat{f}$ as the integral point closest to f in l_2 norm. It is not difficult to see that for every $c \in \mathcal{P}_\epsilon$, $\min\{cs : s \in P(\mathcal{E})\}$ is lower bounded by $\epsilon|\hat{f} - f|$, and this is achieved when the ensemble has the ray $(\hat{f} - f)/|\hat{f} - f|$ with cost ϵ. We prove that this lower bound also holds for minimizing over $G(\mathcal{E})$ instead of $P(\mathcal{E})$. In addition, we show that for most ensembles $\mathcal{E}$, there are enough rays similar to $\hat{f} - f$ that have small cost. This allows us to upper bound $\min\{cs : s \in P(\mathcal{E})\}$ by roughly $\epsilon|\hat{f} - f|$ for most of the ensembles, which gives the desired result.

We start by proving the upper bound. For that, we need to study a specific subset of the ensembles in $\mathcal{D}_n^m$. We remark that the bounds presented are not optimized and were simplified in order to allow a clearer presentation.

4.1 (β, k)-Good Ensembles

Consider an ensemble $\mathcal{E} = \langle f, r^1, \ldots, r^n \rangle$. At a high level, we consider special regions in $\mathbb{R}^m$ 'around' $f - \hat{f}$, whose size depends on a parameter $\beta > 0$; then an ensemble is (β, k)-good if it has at least k rays in each of these regions.

To make this precise, let S^{m-1} denote the $(m-1)$-dimensional unit sphere in $\mathbb{R}^m$. Define $t = \hat{f} - f$ and let ρ be a rotation of $\mathbb{R}^m$ which maps $t/|t|$ into e^m. Let $\bar{C}(\beta)$ be the cap of the hypersphere S^{m-1} consisting of all unit vectors with dot product at least β with e^m. We also define H_i^+ as the halfspace given by $\{x \in \mathbb{R}^m : x_i \geq 0\}$ and $H_i^- = \{x \in \mathbb{R}^m : x_i \leq 0\}$. We use the halfspaces H_i^+ and H_i^- to partition $\bar{C}(\beta)$ into 2^{m-1} parts. That is, for $I \subseteq [m-1]$, let $\bar{C}_I(\beta) = \bar{C}(\beta) \cap (\bigcap_{i \in I} H_i^+) \cap (\bigcap_{i \in [m-1] \setminus I} H_i^-)$. Finally, let $C(\beta) = \rho^{-1}\bar{C}(\beta)$ and $C_I(\beta) = \rho^{-1}\bar{C}_I(\beta)$, that is, the sets obtained by applying the inverse rotation ρ^{-1}.

Using these structures, we say that $\mathcal{E}$ is (β, k)-good if for every $I \subseteq [m-1]$ there are at least k rays r^i in $C_I(\beta)$. The main property of such ensembles is that they allow us to use the following lemma.

Lemma 5. *Let R be a subset of the rays of $\mathcal{E}$ such that $R \cap C_I(\beta) \neq \emptyset$ for all $I \subseteq [m-1]$. Then there is a solution $s \in P(\mathcal{E})$ supported in R such that $\sum_{i=1}^n s_i \leq \frac{|t|}{\beta}$.*

Proof. Without loss of generality assume that $R \cap C(\beta) = \{r^1, r^2, \ldots, r^{n'}\}$. First, one can use Farkas' Lemma and the hypothesis $R \cap C_I(\beta) \neq \emptyset$ for all $I \subseteq [m-1]$ to show that $t \in cone(R \cap C(\beta))$ (see full version of the paper for more details).

So consider $s_1, s_2, \ldots, s_{n'} \geq 0$ with $\sum_{i=1}^{n'} s_i r^i = t$. We claim that $\sum_{i=1}^{n'} s_i \leq |t|/\beta$. To see this, first notice that by definition of $C(\beta)$ we have $r(t/|t|) \geq \beta$ for all $r \in C(\beta)$. Then multiplying the equation $\sum_{i=1}^{n'} s_i r^i = t$ by t gives $\sum_{i=1}^{n'} s_i \beta |t| \leq \sum_{i=1}^{n'} s_i r^i t = tt = |t|^2$ and the claim follows.

Since $f + t = \hat{f}$ is integral we obtain that s is a feasible solution for $P(\mathcal{E})$. This concludes the proof of the lemma.

Using this lemma we can prove an upper bound on optimizing a cost vector in $\mathcal{P}_\epsilon$ over $P(\mathcal{E})$.

Lemma 6. *Fix $\beta, \epsilon > 0$ and an integer $k \geq 0$. Consider a (β, k)-good ensemble $\mathcal{E}$ and let $z(c) = \min\{cs : s \in P(\mathcal{E})\}$. Then*

$$\mathbb{E}_{c \sim \mathcal{P}_\epsilon}[z(c)] \leq |t| \left(p \frac{\epsilon}{\beta^2} + (1-p)\frac{1}{\beta} \right),$$

where

$$p = 1 - 2^{m-1} \left(\frac{1 - \epsilon/\beta}{1 - \epsilon} \right)^k.$$

Proof. Consider a vector c which satisfies the following property: (*) for each $I \subseteq [m-1]$ there is a ray in $C_I(\beta)$ which has cost w.r.t c at most ϵ/β. Then employing Lemma 5 we obtain that $z(c) \leq |t|\epsilon/\beta^2$. Similarly, for a general vector $c \in [\epsilon, 1]^m$ we have the bound $z(c) \leq |t|/\beta$.

Now consider a vector $c \sim \mathcal{P}_\epsilon$. For a fixed I, the probability that every ray in $\mathcal{E} \cap C_I(\beta)$ has cost greater than ϵ/β is at most $((1 - \epsilon/\beta)/(1 - \epsilon))^k$. By union bound, c satisfies property (*) with probability at least

$$1 - 2^{m-1} \left(\frac{1 - \epsilon/\beta}{1 - \epsilon} \right)^k.$$

The lemma then follows by employing the bounds on $z(c)$.

4.2 Probability of Obtaining a (β, k)-Good Ensemble

In this section we estimate the probability that a random ensemble in $\mathcal{D}_n^m$ is (β, k)-good. Let

$$\bar{k} \doteq n \frac{area(\bar{C}_\emptyset(\beta))}{area(S^{m-1})} - \sqrt{\frac{n(\ln n + m - 1)}{2}}. \tag{6}$$

Using some Chernoff bound arguments, we can show the following lemma.

Lemma 7. *Consider a random ensemble $\mathcal{E} \sim \mathcal{D}_n^m$ and let $\bar{k}$ be defined as in (6). If $\bar{k} \geq 0$, then*

$$\Pr\left(\mathcal{E} \text{ is } (\beta, \bar{k})\text{-good}\right) \geq 1 - \frac{1}{n}.$$

4.3 Lower Bound for Simple Splits

In this section we show that $\epsilon|t|$ is also a lower bound for optimizing any vector in $[\epsilon, 1]^n$ over $G(\mathcal{E})$.

Lemma 8. *Fix $\epsilon > 0$ and consider an ensemble $\mathcal{E}$ in $\mathcal{D}_n^m$ and a vector $c \in [\epsilon, 1]^n$. For t defined as before, we have*

$$\min\{cs : s \in G(\mathcal{E})\} \geq \epsilon|t|.$$

Proof. To prove this lemma, let $S_i \equiv \sum_{j=1}^n \psi^i(r^j)s^j \geq 1$ be the inequality for $P(\mathcal{E})$ obtained from the simple split $\{x : 0 \leq x_i \leq 1\}$. Clearly S_i is valid for $G(\mathcal{E})$. Using the definition of Minkowski's functional, it is not difficult to see that

$$\psi^i(r^j) = \frac{r_i^j}{[r_i^j \geq 0] - f_i},$$

where $[r_i^j \geq 0]$ is the function that is equal to 1 if $r_i^j \geq 0$ and equal to 0 otherwise.

Now consider the inequality $\sum_{j=1}^n \psi(r^j)s_j \geq 1$ where

$$\psi(r^j) = \frac{\sum_{i=1}^m (\hat{f}_i - f_i)^2 \psi^i(r^j)}{\sum_{i=1}^m (\hat{f}_i - f_i)^2}.$$

This inequality is a non-negative combination of the inequalities S_i and therefore is valid for $G(\mathcal{E})$. We claim that for any $c \in [\epsilon, 1]^m$, $\min\{cs : \sum_{j=1}^n \psi(r^j)s_j \geq 1\} \geq \epsilon|t|$, which will give the desired lower bound on optimizing c over $G(\mathcal{E})$.

To prove the claim recall that $\sum_{i=1}^m (\hat{f}_i - f_i)^2 = |t|^2$ and notice that

$$\psi(r^j) = \frac{1}{|t|^2} \sum_{i=1}^m (\hat{f}_i - f_i)^2 \psi^i(r^j) = \frac{1}{|t|^2} \sum_{i=1}^m \frac{(\hat{f}_i - f_i)^2 r_i^j}{[r_i^j \geq 0] - f_i}.$$

Employing the Cauchy-Schwarz inequality and using the fact that $|r^j| = 1$, we get

$$\psi(r^j) \leq \frac{1}{|t|^2} |r^j| \sqrt{\sum_{i=1}^m \left(\frac{(\hat{f}_i - f_i)^2}{[r_i^j \geq 0] - f_i} \right)^2} \leq \frac{1}{|t|^2} \sqrt{\sum_{i=1}^m \frac{(\hat{f}_i - f_i)^4}{([r_i^j \geq 0] - f_i)^2}}.$$

However, since $\hat{f}$ is the integral point closest to f, for all i it holds that $(\hat{f}_i - f_i)^2 \leq ([r_i^j \geq 0] - f_i)^2$. Employing this observation on the previous displayed inequality gives $\psi(r^j) \leq 1/|t|$. Therefore, any s satisfying $\sum_{j=1}^n \psi(r^j)s_j \geq 1$ also satisfies $\sum_{j=1}^n s_j \geq |t|$. The claim then follows from the fact that every coordinate of c is lower bounded by ϵ. This concludes the proof of Lemma 8.

4.4 Proof of Theorem 2

Recall that ϵ, α and m are fixed. Let β be the minimum between $\sqrt{2/\alpha}$ and a positive constant strictly less than 1; this guarantees that $\bar{C}_\emptyset(\beta) > 0$. Consider

a large enough positive integer n. Let $\mathcal{E}$ be a $(\beta, \bar{k})$-good ensemble in $\mathcal{D}_n^m$, where $\bar{k}$ is defined as in (6). Notice that $\bar{k}$, as a function of n, has asymptotic behavior $\Omega(n)$. We assume that n is large enough so that $\bar{k} > 0$.

Now let us consider Lemma 6 with $k = \bar{k}$. The value p defined in this lemma is also function of n, now with asymptotic behavior $1-o(1)$. Thus, if n is chosen sufficiently large we get $1-p \leq \epsilon\beta\alpha/2$ and hence $\mathbb{E}_{c\sim\mathcal{P}_\epsilon}[z(c)] \leq |t|\epsilon\alpha$. If in addition we use the lower bound from Lemma 8, we obtain that $\text{avg}(P(\mathcal{E}), G(\mathcal{E}), \mathcal{P}_\epsilon) \leq \alpha$. The theorem then follows from the fact that an ensemble in $\mathcal{D}_n^m$ is $(\beta, \bar{k})$-good with probability at least $1 - 1/n$, according to Lemma 7.

References

1. Andersen, K., Louveaux, Q., Weismantel, R., Wolsey, L.A.: Inequalities from two rows of a simplex tableau. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 1–15. Springer, Heidelberg (2007)
2. Balas, E.: Intersection cuts-a new type of cutting planes for integer programming. Operations Research 19(1), 19–39 (1971)
3. Balas, E., Qualizza, A.: Personal communication
4. Balas, E., Saxena, A.: Optimizing over the split closure. Mathematical Programming, Series A 113(2), 219–240 (2008)
5. Basu, A., Bonami, P., Cornuéjols, G., Margot, F.: Experiments with two-row cuts from degenerate tableaux. To appear in INFORMS Journal on Computing, doi:10.1287/ijoc.1100.0437
6. Basu, A., Bonami, P., Cornuéjols, G., Margot, F.: On the relative strength of split, triangle and quadrilateral cuts. Mathematical Programming, Series A 126, 281–314 (2011)
7. Basu, A., Cornuéjols, G., Molinaro, M.: A probabilistic comparison of split, triangle and quadrilateral cuts for two row mixed-integer programs (extended version), `http://www.optimization-online.org/DB_HTML/2010/10/2770.html`
8. Borozan, V., Cornuéjols, G.: Minimal valid inequalities for integer constraints. Mathematics of Operations Research 34(3), 538–546 (2009)
9. Cornuéjols, G., Margot, F.: On the facets of mixed integer programs with two integer variables and two constraints. Mathematical Programming, Series A 120, 429–456 (2009)
10. Dey, S.S., Lodi, A., Tramontani, A., Wolsey, L.A.: Experiments with two row tableau cuts. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 424–437. Springer, Heidelberg (2010)
11. Espinoza, D.G.: Computing with multi-row gomory cuts. Operations Research Letters 38(2), 115–120 (2010)
12. Gomory, R.E.: Some polyhedra related to combinatorial problems. Linear Algebra and its Applications 2(4), 451–558 (1969)
13. He, Q., Ahmed, S., Nemhauser, G.: A probabilistic comparison of split and type 1 triangle cuts for two row mixed-integer programs. Working paper, School of OR&IE, Goergia Tech (2009), `http://www.optimization-online.org/DB_HTML/2010/06/2636.html`
14. Del Pia, A., Wagner, C., Weismantel, R.: A probabilistic comparison of the strength of split, triangle, and quadrilateral cuts. Working paper, ETH Zürich (September 2010), `http://arxiv.org/abs/1009.5253`

Partial Convexification of General MIPs by Dantzig-Wolfe Reformulation

Martin Bergner[1,⋆], Alberto Caprara[2], Fabio Furini[1], Marco E. Lübbecke[1], Enrico Malaguti[2], and Emiliano Traversi[2]

[1] Chair of Operations Research, RWTH Aachen University, Templergraben 64, 52056 Aachen, Germany
{martin.bergner,fabio.furini,marco.luebbecke}@rwth-aachen.de
[2] DEIS, Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
{alberto.caprara,enrico.malaguti,emiliano.traversi2}@unibo.it

Abstract. Dantzig-Wolfe decomposition is well-known to provide strong dual bounds for specially structured mixed integer programs (MIPs) in practice. However, the method is not implemented in any state-of-the-art MIP solver: it needs tailoring to the particular problem; the decomposition must be determined from the typical bordered block-diagonal matrix structure; the resulting column generation subproblems must be solved efficiently; etc. We provide a computational proof-of-concept that the process can be automated in principle, and that strong dual bounds can be obtained on general MIPs for which a solution by a decomposition has not been the first choice. We perform an extensive computational study on the 0-1 dynamic knapsack problem (without block-diagonal structure) and on general MIPLIB2003 instances. Our results support that Dantzig-Wolfe reformulation may hold more promise as a general-purpose tool than previously acknowledged by the research community.

1 Introduction

A considerable, if not the major, part of the computational (mixed) integer programming machinery is about outer approximating the convex hull of integer feasible points (or mixed integer sets). The addition of valid inequalities, a.k.a. cutting planes, is the traditional general-purpose device which proved powerful in strengthening the linear programming relaxations. Given that the integer hull is the ultimate object of desire, we ask: Why don't we just work with it? Being fully aware of the boldness of this question, we want to seriously re-consider it by *explicitly* constructing parts of the integer hull via a generic Dantzig-Wolfe type reformulation (decomposition). This extends previous *partial convexification* approaches which only separate a subset of facets from the integer hull.

Dantzig-Wolfe reformulation (DWR) of mixed integer programs (MIPs) became a computationally very successful—sometimes the only applicable—approach to

⋆ Supported by the German Research Foundation (DFG) as part of the Priority Program "Algorithm Engineering" under grant no. LU770/4-1.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 39–51, 2011.

producing high-quality solutions for well-structured combinatorial optimization problems like vehicle routing, cutting stock, p-median, generalized assignment, and many others. Their common structure is the (bordered) block-diagonal form of the coefficient matrix, the traditional realm of Dantzig-Wolfe decomposition. Be aware that so far its use is tailored to the application and far from being a general-purpose tool: It is the *user* who does not only know *that* there is an exploitable structure present but also *what* it looks like, and *how* to exploit it algorithmically. In particular in view of the automatism with which general-purpose cutting planes are separated in all serious MIP solvers, this is an unsatisfactory situation. This raises several research questions of increasing ambition:

- When the MIP contains a *known structure* suitable to Dantzig-Wolfe reformulation, can *an algorithm* detect and exploit it?
- When the contained structure is *unknown* (to be stated more precisely later), can it still be detected and exploited?
- When it is known that the MIP *does not contain* a structure amenable to DWR in the traditional sense, can DWR still be a useful computational tool?

Re-arranging matrices into particular forms is a well-known topic. However, as we will see, when there are several choices, a "good" one may not be obvious to find at all. Besides our work, we are not aware of any attempts to systematically answer the first two questions in a DWR context, and it can be taken as a fact that the research community is very inclined to answer the last, and most interesting question in the negative. In our computational study on several of the hardest MIPLIB2003 instances, we do not only suggest first attempts to accomplish the structure detection in a DWR context. We also support the DWR's potential of becoming a general-purpose method for improving dual bounds by giving surprisingly encouraging computational results. Of course, at the moment, our work is not intended to produce a competitive tool, but to provide a proof-of-concept and demonstrate that the direction is promising. The main findings of our work can be summarized as follows:

- A known or hidden double-bordered block-diagonal (so-called: *arrowhead*) matrix structure can be effectively recovered and prepared for use in DWR by a suitable use of (hyper-)graph partitioning algorithms.
- For the dynamic 0-1 knapsack problem, the natural formulation of which does not expose any bordered block-diagonal structure, we give impressive computational results which demonstrate the potential of our method for closing the integrality gap.
- For some of the hardest MIPLIB2003 instances our reformulations produce stronger dual bounds than `CPLEX` 12.2 with default cutting planes enabled.
- We provide a general-purpose implementation which reads an LP file and performs the detection, the reformulation, and the column generation itself in order to obtain a strong LP relaxation, with only mild user interaction.

The flow of the paper is as follows: We briefly introduce the concept of partial convexification, and the overall approach. This is then applied first to a problem

not containing a matrix structure directly amenable to classical Dantzig-Wolfe reformulation, and then to general MIPs. We report on extensive computational experiments and close with a discussion of our results.

1.1 Partial Convexification and Dantzig-Wolfe Reformulations

Consider a MIP of the form

$$\max\{c^t x : Ax \leq b,\ Dx \leq e,\ x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}\ . \tag{1}$$

Let $P := \{x \in \mathbb{Q}^n : Dx \leq e\}$ and $P_{IP} := \text{conv}\{P \cap \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}$ denote the LP relaxation and the integer hull with respect to constraints $Dx \leq e$, respectively. We assume for ease of presentation that P is bounded. In the classical *Dantzig-Wolfe reformulation* of constraints $Dx \leq e$ we express $x \in P_{IP}$ as a convex combination of the vertices V of P_{IP}, which leads to

$$\max\{c^t x : Ax \leq b,\ x = \sum_{v \in V} \lambda_v v,\ \sum_{v \in V} \lambda_v = 1,\ \lambda \geq 0,\ x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}\ . \tag{2}$$

It is well-known that the resulting LP relaxation is potentially stronger than that of (1) when $P_{IP} \subsetneq P$, which is a main motivation of performing the reformulation in the first place. This *partial convexification* with respect to the constraints $Dx \leq e$ corresponds to adding (implicitly) *all* valid inequalities for P_{IP} to (1), which in a sense is the best one can hope for.

The reformulated MIP (2) has fewer constraints remaining, the so-called *master constraints* $Ax \leq b$, plus the convexity constraint and the constraints linking the *original* x variables to the *extended* λ variables. On the downside of it, in general MIP (2) has an exponential number of λ variables, so its LP relaxation is solved by column generation, where the pricing or *slave MIP* problem to check whether there are variables with positive reduced cost to be added to the current *master LP* problem calls for the optimization of a linear objective function over P_{IP}. This slave MIP can either be solved by a general-purpose solver or by a tailored algorithm to exploit a specific structure, if known.

In the classical DWR setting, k disjoint sets of constraints are partially convexified, namely when the matrix D has block-diagonal form

$$D = \begin{bmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^k \end{bmatrix},$$

where $D^i \in \mathbb{Q}^{m_i \times n_i}$ for $i = 1, \ldots, k$. In other words, $Dx \leq e$ decomposes into $D^i x^i \leq e^i$ $(i = 1, \ldots, k)$, where $x = (x^1, x^2, \ldots, x^k)$, with x^i being an n_i-vector for $i = 1, \ldots, k$. Every $D^i x^i \leq e^i$ individually is partially convexified in the above spirit. We call k the *number of blocks* of the reformulation. Often enough,

constraints are not separable by variable sets as above, and a double-bordered block-diagonal (or *arrowhead*) form is the most specific structure we can hope for, i.e. the constraint matrix of (1) looks like this

$$\begin{bmatrix} D^1 & & & & F^1 \\ & D^2 & & & F^2 \\ & & \ddots & & \vdots \\ & & & D^k & F^k \\ A^1 & A^2 & \cdots & A^k & G \end{bmatrix}.$$

The constraints associated with the rows of A^1 are called the *coupling constraints* and the variables associated with the columns of F^1 are called the *linking variables*. One can obtain a form without linking variables (and with additional linking constraints) by replacing each linking variable by one copy for each nonzero entry of the associated column and adding constraints imposing that all these copies be equal (see e.g., [13]). Then we are back to the above traditional setting.

1.2 Related Literature

For a general background on Dantzig-Wolfe reformulation of MIPs we refer to the recent survey [15]. The notion of *partial convexification* has been used before. For instance, Sherali et al. [12] choose small subsets of integer variables (and usually only one constraint) to directly separate cutting planes from the partial convexification P_{IP}. Our approach goes far beyond that in terms of number of constraints and variables involved in the reformulation.

There are several implementations which perform a Dantzig-Wolfe reformulation of a general MIP, and handle the resulting column generation subproblems in a generic way, like `BaPCod` [14], `DIP` [11], `G12` [10], and `GCG` [8]. In [8] it is shown that a generic reformulation algorithm performs well when a block-diagonal matrix structure is *known and given* to the algorithm. Tebboth, in his thesis [13], derived some decomposable matrix structures from the problem given in a specific modeling language. A similar approach is taken in the `G12` project. However, we do not know of a code which automatically detects a possible structure just from the matrix, that is well-suited and helpful for a Dantzig-Wolfe reformulation (or creates one by variable splitting), let alone evaluates or exploits it.

Bordered block-diagonal matrices play an important role in, e.g., numerical linear algebra. One typically tries to identify a fixed number of blocks of almost equal size with as few constraints in the border as possible. The motivation is to prepare a matrix for parallel computation, like for solving linear equation systems, see, e.g., [2], and the references therein. We are not aware of any attempts to evaluate the quality of the re-arranged matrices in terms of suitability for DWR.

We would also like to note the following possible connection to multi-row cuts which recently received some attention. Computational experiments [6] suggest that obtaining valid inequalities from more than one row of the simplex tableau holds some potential. However, in order to use this potential it seems to be

imperative to have a criterion for which rows to select. As our choice of which rows to convexify is equally important for similar reasons, the lessons we learn may help there as well.

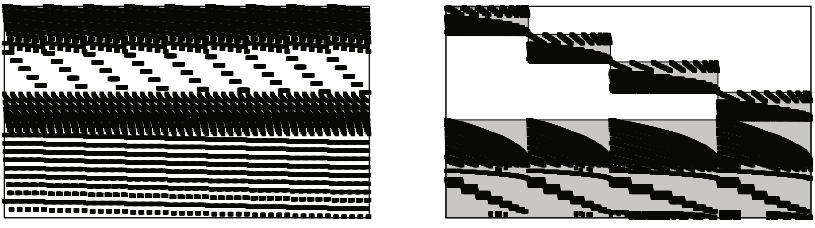
2 Almost Automatic Detection of an Arrowhead Form

As mentioned, permuting a matrix A into arrowhead form is a common topic in numerical linear algebra. There is a folklore connection between a matrix and an associated (hyper-)graph which is also used in the graph partitioning approach by Ferris and Horn [7]. We first briefly illustrate their algorithm and then adapt it to our purpose of finding tighter LP relaxations of MIPs through DWR.

Assume the number k of blocks is given. Consider the bipartite graph G with one node r_i associated with each row i of A, one node c_j associated with each column j of A, and one edge (r_i, c_j) whenever $a_{ij} \neq 0$. Assuming the number $m+n$ of nodes is a multiple of k, find a *min k-equicut* of G, i.e. partition its node set into k subsets of equal size so that the number of edges in the cut, i.e. that join nodes in different subsets, is minimized. Finally, remove a set of nodes of G so as to ensure, for the remaining graph, that no edge joins nodes in different subsets of the partition. This yields the final arrowhead form: the row nodes removed represent the coupling constraints, the column nodes removed represent the linking variables, and the remaining row and column nodes in the ith subset of the partition define the submatrix D^i. In practice, min k-equicut is solved heuristically by using `Metis` which is an implementation of the multilevel graph partitioning algorithm in [9]. The final removal is done greedily by iteratively removing the node joined to nodes in other subsets by the largest number of edges. Moreover, to eliminate the need to require that the number of nodes be a multiple of k or to insist in having subsets of the same size, dummy nodes disconnected from the rest of the graph are added before doing the partitioning. This imposes only an upper bound on the size of each subset and on the number of subsets (note that subsets with dummy nodes only are irrelevant in the end).

We use a variant of this method because we would like to avoid the final removal phase. Specifically, we define the hypergraph H with a *node* for each nonzero entry of A. There is a *constraint hyperedge* joining all nodes for nonzero entries in the ith row of A. Moreover, there is a *variable hyperedge* joining all nodes for nonzero entries in the jth column of A. To each hyperedge we associate a cost to penalize the use of coupling constraints and linking variables.

Then, we find a heuristic min k-equicut of H, now with the objective of minimizing the sum of the costs of the hyperedges in the cut, i.e. that join nodes in different subsets of the partition. The solution already defines the arrowhead form, as the constraint hyperedges in the cut represent the coupling constraints, the variable hyperedges in the cut represent the linking variables, and the remaining constraint and variable hyperedges joining only nodes in the ith subset of the partition define the submatrix D^i. Also in this case we use dummy nodes for allowing for unequal block sizes, 20% proved useful.



(a) original *10teams* (b) detected structure

Fig. 1. (a) Matrix structure directly from the LP file (*10teams*) and (b) with a bordered block-diagonal structure detected by our algorithm

3 Computational Results

3.1 Notes on the Implementation and Experimental Setup

All experiments were done on Intel i7 quad-core PCs (2.67MHz, 8GB memory) running Linux 2.6.34 (single thread). As MIP and LP solver we used `CPLEX` 12.2 (single thread). Two independent jobs were run in parallel which has an impact on the CPU time of about 3% in our experiments. The min k-equicut problems to determine a reformulation (see Sect. 2) are heuristically solved using `Hmetis` [9].

Concerning the implementation, the overall algorithm detects an arrowhead form in the matrix, splits linking variables in order to obtain a bordered block-diagonal form, and then performs a Dantzig-Wolfe reformulation of the resulting blocks. Certain parameters must be given to the algorithm like the number k of blocks or the weights of the hyperedges in the k-equicut problem. We experimented with very few different settings and selected good decompositions instance dependent to demonstrate the concept. In the full version of this paper we plan to show that the whole process can be automated without any user interaction at all.

3.2 A Class of Structured MIPs without Block-Diagonal Form

We first consider a class of problems which has a natural MIP formulation with a coefficient matrix which is very structured, but not bordered block-diagonal at all. A *dynamic MIP* extends a MIP

$$\max\{c^t x : Ax \leq b,\ x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}$$

by a time horizon $[0, T]$ and a time interval $[s_j, f_j] \subseteq [0, T]$ in which each variable x_j is *active*. In the dynamic MIP all constraints must be satisfied at any instant in the time horizon restricting attention to the variables that are active at that instant (with the other variables fixed to 0). It is elementary to observe that attention can be restricted to the instants $I := \{s_1, \ldots, s_n, f_1, \ldots, f_n\}$ in which variables become active or inactive, yielding the following MIP formulation of the dynamic MIP:

$$\max\{c^t x : A^i x \leq b\ (i \in I),\ x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}, \tag{3}$$

where A^i is obtained from A by setting to 0 all the entries of columns associated with variables not active at $i \in I$.

A simple example of a dynamic MIP is the *dynamic 0-1 knapsack* problem, in which $q = 0$ and $Ax \leq b$ takes the form $a^t x \leq b$, $0 \leq x \leq 1$ for some $a \in \mathbb{Q}_+^n$ and $b \in \mathbb{Q}_+$. In words, we have a set of items each with a size a_j and active for time interval $[s_j, f_j]$ and a container of capacity b and we must select a set of items to pack in the container (for the whole interval in which they are active) so that the total size packed in each instant is at most b. This problem is also known as *resource allocation* or *unsplittable flow on a line*. The latter name is due to the fact that the time dimension may in fact be a (one-dimensional) spatial dimension, as it is the case in the following real-world application that we encountered in railway service design [5]. We are given a railway corridor with m stations $1, \ldots, m$ and n railcars, the jth having weight a_j and to be transported from station s_j to station f_j, gaining profit c_j in this case. Moreover, we have a train that travels from station 1 to station m and can carry at the same time railcars for a total weight b. The problem is to decide which railcars the train carries in order to maximize the profit, and is clearly a dynamic 0-1 knapsack problem.

The dynamic 0-1 knapsack problem was recently proven to be strongly NP-hard and approximable within a constant [3] but a polynomial-time approximation scheme is not known.

In Table 1 we report the results obtained on a hard representative instance in each of the 0-1 dynamic knapsack classes considered in [4] when the parameter k varies. This parameter is chosen in such a way to enforce a decomposition in blocks of equal size of 128, 64, and 32 constraints each (this proved to provide best bounds as compared to computation times; for a larger number of much smaller blocks, computation times went up considerably). As one can see, on average, the DWR approach is able to close almost all of the root node's integrality gap. However, it takes an order of magnitude longer to obtain these results (see below). Nonetheless, the excellent quality of the dual bound helps in faster obtaining an integer optimal solution, as we see next.

Even though we do not wish to advocate for a principal "battle" of branch-and-price "versus" branch-and-cut (because branch-price-and-cut is probably the most versatile computational tool anyway), the direct comparison is instructive. To this end, we developed a very basic depth-first branch-and-price algorithm. Each node's linear programming relaxation is solved via DWR. We branch on the most fractional *original* variable and compare this setting to `CPLEX` 12.2 default branch-and-cut. Table 2 lists the results. We report the time required by branch-and-bound and, in case this exceeds the time limit of one hour, the relative gap between the best dual bound and the best primal solution found. The table shows the clear advantage of using the DWR. All instances can be solved to optimality within minutes, whereas `CPLEX` reaches the one-hour time limit for all instances except I65 (where profits equal volume). The results on the other instances in the six classes are analogous.

Table 1. Quality of the dual bound obtained by our DWR approach on the dynamic 0-1 knapsack problem, in comparison to a state-of-the-art IP solver. Listed are the instance name, the number of constraints and variables, the number k of blocks, the number ℓ of linking variables, and number c of coupling constraints. Under the heading *LP* one finds the relative integrality gap of the LP relaxation (in percent). The relative integrality gaps of DWR and `CPLEX` with default cuts applied are listed under *DWR* and *`CPLEX`+cuts*, respectively. The percentage of the LP gap closed is given under *%closed* for both approaches. The last row lists arithmetic means of the columns.

						LP	DWR		`CPLEX`+cuts	
instance	rows	cols	k	ℓ	c	gap	gap	%closed	gap	%closed
I45	1280	3433	10	243	0	15.720	0.002	**99.987**	1.306	91.695
I55	1280	8266	10	189	0	13.944	0.000	**100.000**	0.638	95.421
I65	1280	3434	10	243	0	11.182	0.011	**99.903**	0.098	99.127
I75	1280	4771	10	199	0	13.783	0.026	**99.808**	0.796	94.228
I85	1280	8656	10	159	0	13.057	0.001	**99.994**	0.410	96.860
I95	1280	5209	10	243	0	12.306	0.000	**100.000**	0.756	93.860
I45	1280	3433	20	505	0	15.720	0.007	**99.956**	1.306	91.695
I55	1280	8266	20	402	0	13.944	0.029	**99.793**	0.638	95.421
I65	1280	3434	20	509	0	11.182	0.009	**99.915**	0.098	99.127
I75	1280	4771	20	430	0	13.783	0.009	**99.934**	0.796	94.228
I85	1280	8656	20	344	0	13.057	0.001	**99.994**	0.410	96.860
I95	1280	5209	20	515	0	12.306	0.000	**100.000**	0.756	93.860
I45	1280	3433	40	977	0	15.720	0.047	**99.703**	1.306	91.695
I55	1280	8266	40	825	0	13.944	0.020	**99.857**	0.638	95.421
I65	1280	3434	40	976	0	11.182	0.000	**100.000**	0.098	99.127
I75	1280	4771	40	880	1	13.783	0.031	**99.775**	0.796	94.228
I85	1280	8656	40	756	0	13.057	0.025	**99.807**	0.410	96.860
I95	1280	5209	40	1041	0	12.306	0.004	**99.970**	0.756	93.860
means						13.332	0.012	**99.911**	0.667	95.199

3.3 MIPLIB2003

In order to assess the generality of the proposed method we tested the algorithm on MIPLIB2003 instances [1]. We selected a subset of instances, for which (a) the optimum is known, (b) the density is between 0.05% and 5%, (c) the number of non-zeros is not larger than 20,000, and (d) the percentage of discrete variables is at least 20%. We are consistently capable of improving over the LP relaxation in terms of the dual bound. Moreover, on average, our DWR approach closes a larger percentage of the integrality gap than `CPLEX` with default cuts applied, see Table 3. We do not report on computation times because the experience is the same as for the RAP (and not the focus here): Often, we need an order of magnitude more time (even though occasionally, we are competitive with `CPLEX`).

Table 2. Comparison of the dual bounds obtainable by branch-and-price as compared to branch-and-cut for the dynamic 0-1 knapsack problem, and the computation times needed. All instances are solved to optimality by branch-and-price. Listed are the instance name, the times to obtain the various dual bounds: The linear relaxation from DWR, the linear relaxation with CPLEX and default cuts applied, the optimal solution with branch-and-price (DWR), and the time needed with branch-and-cut (CPLEX). The time limit (TL) was set to one hour. The final columns list the remaining integrality gap of CPLEX at time limit.

	time				CPLEX B&B	
instance	DWR	CPLEX+cuts	DWR B&P	CPLEX B&B	gap	%closed
I45	33.75	2.72	**111.06**	TL	0.35	97.79
I55	179.73	4.53	**2483.31**	TL	0.16	98.87
I65	7.31	1.50	16.28	**1.67**	0.00	100.00
I75	30.02	2.63	**97.84**	TL	0.03	99.80
I85	55.62	4.53	**105.59**	1357.88	0.00	100.00
I95	29.21	3.01	**29.22**	TL	0.05	99.55
means	55.94	3.15	**473.88**	2626.59	0.10	99.34

Choosing a good decomposition. We experimented with a few parameter settings for our algorithm to detect an arrowhead structure. Depending on these settings, different arrowhead forms are produced (for the same input matrix), and these perform differently in the subsequent Dantzig-Wolfe reformulation. Parameters are (a) the number k of blocks, and the penalties for hyperedges which (b) split continuous and (c) discrete variables, as well as (d) penalties for hyperedges which couple constraints. We experimented with $k \in \{2, 3, 4, 5\}$, penalized (throughout all experiments) hyperedges corresponding to continuous and discrete with cost 1 and 2, respectively; and finally we used two different settings for penalizing coupling constraints: mildly (cost 5) and extremely (cost 10^5). Every combination was applied to each instance, and the best result in terms of dual bound is reported. In other words, in this paper, we give only a proof-of-concept *that* a good decomposition can be chosen. *How* to find a good decomposition, and even the meaning of "good," are highly non-trivial issues. E.g., the "right" number k of blocks is far from obvious for instances that do not have a natural (bordered) block-diagonal form. We have preliminary computational experience that a "visually appealing" decomposition performs better than others. We give details on measures for this intuition, and on how to automate the detection and decomposition process in the full version.

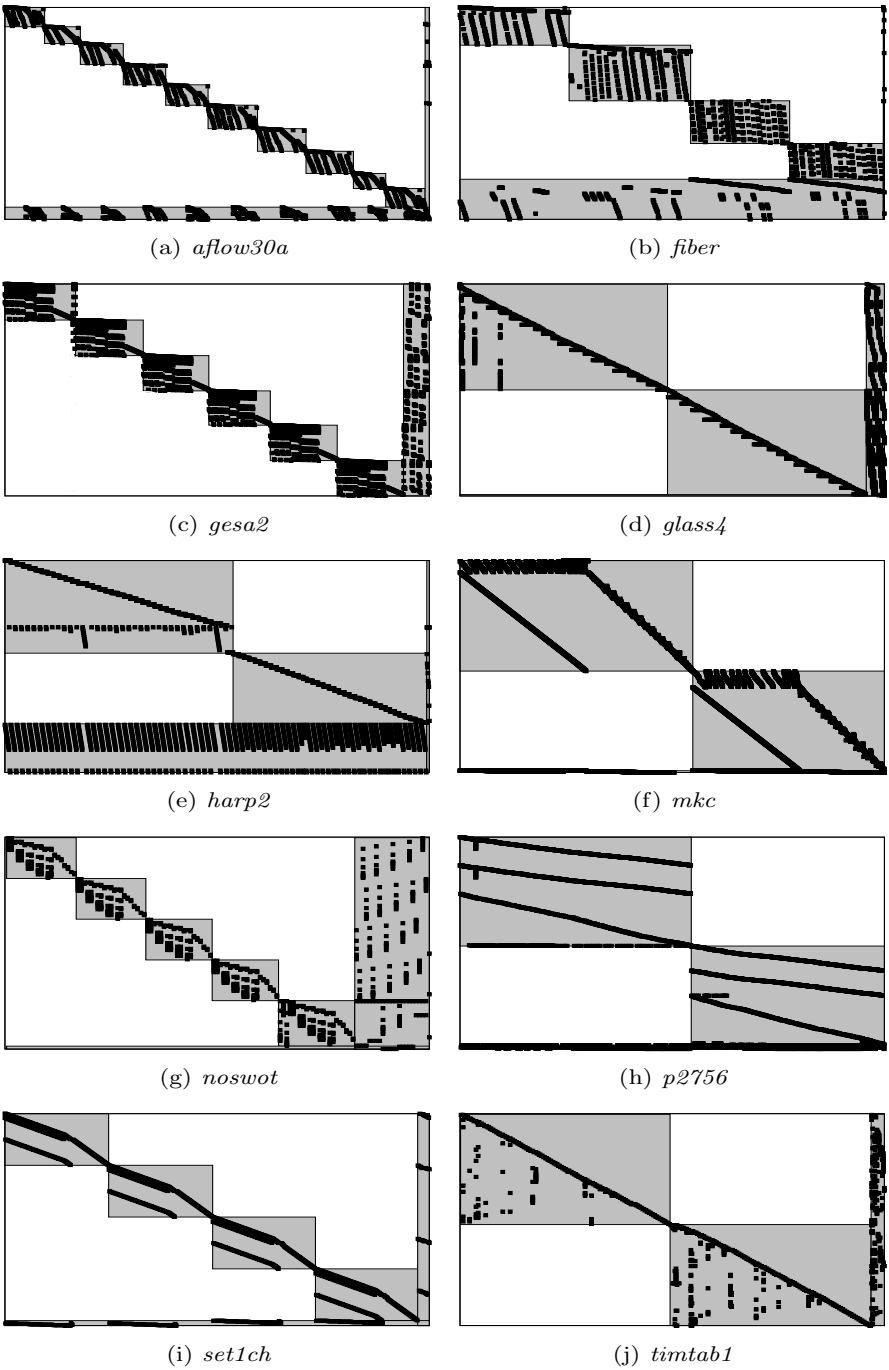
Table 3 shows that in almost all cases the dual bound found by our DWR approach is much better than that of the continuous relaxation, and often even improves on CPLEX's root node bounds with default cuts applied. The time required to compute our bound is not competitive with the time required by the general-purpose solver to solve the instance, but there remains the possibility that for some instances the significantly stronger dual bound helps in solving the instance to integer optimality.

Table 3. Comparison of the dual bounds provided by our automatic DWR reformulation approach and the general-purpose MIP solver CPLEX for 23 selected instances of MIPLIB2003. The headings have the same meanings as in Table 1.

						LP	DWR		CPLEX+cuts	
instance	rows	cols	k	ℓ	c	gap	gap	%closed	gap	%closed
10teams	2025	230	4	0	107	0.758	0.000	**100.000**	0.000	**100.000**
aflow30a	842	479	2	0	28	15.098	14.700	2.634	5.353	**64.547**
aflow40b	2728	1442	5	0	39	13.899	13.899	0.000	6.471	**53.441**
fiber	1298	363	2	2	21	61.550	1.067	**98.266**	1.894	96.923
fixnet6	878	478	4	3	14	69.850	18.882	72.967	6.064	**91.318**
gesa2-o	1224	1248	5	65	0	1.177	0.000	**99.986**	0.207	82.379
gesa2	1224	1392	3	65	0	1.177	0.000	**99.986**	0.100	91.507
glass4	322	396	3	16	0	33.334	26.713	**19.862**	33.334	0.000
harp2	2993	112	5	0	39	0.614	0.614	0.000	0.371	**39.599**
manna81	3321	6480	2	78	0	1.010	0.000	**100.000**	0.000	**100.000**
mkc	5325	3411	2	0	29	8.514	0.153	**98.200**	3.778	55.625
modglob	422	291	2	18	3	1.493	0.000	**100.000**	0.142	90.480
noswot	128	182	5	21	3	4.878	0.488	**90.000**	4.878	0.000
opt1217	769	64	4	0	16	25.134	25.134	0.000	0.000	**100.000**
p2756	2756	755	4	39	13	13.932	0.269	**98.070**	7.467	46.407
pp08a	240	136	2	16	0	62.608	2.172	**96.530**	2.525	95.967
pp08aCUTS	240	246	2	16	0	25.434	2.172	**91.459**	3.823	84.968
rout	556	291	5	0	16	8.881	0.681	**92.335**	8.858	0.262
set1ch	712	492	3	20	8	41.311	2.086	94.950	0.923	**97.765**
timtab1	397	171	2	13	0	96.248	14.473	**84.963**	39.050	59.428
timtab2	675	294	4	25	0	92.377	24.426	**73.559**	46.004	50.200
tr12-30	1080	750	3	24	0	89.119	2.713	96.955	0.682	**99.235**
vpm2	378	234	2	7	0	28.078	1.706	**93.924**	6.443	77.053
arithm. mean						30.281	6.624	**74.115**	7.755	**68.570**

4 Discussion

We have performed the first systematic computational study with an automatic partial convexification by a Dantzig-Wolfe type reformulation of subsets of rows of *arbitrary* mixed integer programs. While it is clear from theory that a partial convexification can improve the dual bound, it has not been considered a generally useful computational tool *in practice*. Thus, the most unexpected outcome of our study is that already a fairly basic implementation, combined with a careful choice of the decomposition, is actually capable of competing with or even beating a state-of-the-art MIP solver in terms of the root node dual bound. Interestingly, to the best of our knowledge for the first time, the "careful choice of the decomposition" is done almost entirely by an algorithm, only mildly helped by the user. A fully automated detection will be presented in the full paper.



(a) *aflow30a* (b) *fiber*

(c) *gesa2* (d) *glass4*

(e) *harp2* (f) *mkc*

(g) *noswot* (h) *p2756*

(i) *set1ch* (j) *timtab1*

Fig. 2. Detected matrix structures for selected MIPLIB2003 instances

One should not deny that an inner approximation of the integer hull still has considerable disadvantages, as the process is not reversible: we cannot easily get rid of the extended formulation. Also, the choice of the decomposition is final (at present). This "single shot" contrasts cutting plane algorithms which can iteratively increase the number of classes of valid inequalities considered. Therefore, one must carefully decide whether to use a DWR approach or not. A remote goal would be to be able to make this important decision based on the instance only. If in, say, even only a small fraction of "all" MIPs a DWR approach pays, we would have already enriched the generic MIP toolbox.

There are some possible immediate extensions concerning the implementation. Even though we have mainly solved the LP relaxation so far, our tool is able to perform a true branch-and-price. Only further experimentation can show whether the advantage in the root node can be retained throughout the search tree, not only for dynamic MIPs but also for general MIPs (it is also conceivable that an advantage becomes visible *only* further down the tree). If one is only interested in a strong dual bound, the addition of generic cutting planes is a natural next step.

All the questions initially posed in the introduction are computationally and conceptually extremely hard, and at present one cannot hope for conclusive answers to any of them. We therefore think that our work spawns a number of interesting research directions worth pursuing further.

1. The most important task, both from a theoretical and a practical point of view, is to characterize a *good decomposition*. This can also help in quickly deciding whether it is worth trying a reformulation or not.
2. We have seen that the matrix needs not contain any (known) apparent structure in order to make the method perform well. In particular our third question from the introduction needs re-formulation in the light of our results: what does the fact that a model is suitable for application of DWR mean?
3. *Extended formulations* are a topic on its own in combinatorial optimization, mainly used as a theoretical vehicle to obtain stronger formulations. As DWR is a particular kind of extended formulation it is natural to ask: Can an approach like ours turn this into a computational tool?
4. We complained that, once chosen, a decomposition is static. Is there a computationally viable way for dynamically updating an extended formulation, like our DWR?

Taking into account that state-of-the-art solvers make successful use of cutting planes for over 15 years now, it is clear that outer approximations of the integer hull have a prominent headway in experience over inner approximations. We hope to have inspired further research and experimentation on the topic of this paper.

References

1. Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. Oper. Res. Lett. 34(4), 361–372 (2006)
2. Aykanat, C., Pinar, A., Çatalyürek, Ü.V.: Permuting sparse rectangular matrices into block-diagonal form. SIAM J. Sci. Comput. 25, 1860–1879 (2004)
3. Bonsma, P., Schulz, J., Wiese, A.: A constant factor approximation algorithm for unsplittable flow on paths. CoRR, abs/1102.3643 (2011)
4. Caprara, A., Furini, F., Malaguti, E.: Exact algorithms for the temporal knapsack problem. Technical report OR-10-7, DEIS, University of Bologna (2010)
5. Caprara, A., Malaguti, E., Toth, P.: A freight service design problem for a railway corridor. Tranportation Sci (2011) (in press)
6. Espinoza, D.G.: Computing with multi-row Gomory cuts. Oper. Res. Lett. 38, 115–120 (2010)
7. Ferris, M.C., Horn, J.D.: Partitioning mathematical programs for parallel solution. Math. Program. 80(1), 35–61 (1998)
8. Gamrath, G., Lübbecke, M.E.: Experiments with a generic dantzig-wolfe decomposition for integer programs. In: Festa, P. (ed.) SEA 2010. LNCS, vol. 6049, pp. 239–252. Springer, Heidelberg (2010)
9. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Comput. 20(1), 359–392 (1998)
10. Puchinger, J., Stuckey, P.J., Wallace, M.G., Brand, S.: Dantzig-Wolfe decomposition and branch-and-price solving in G12. Constraints 16(1), 77–99 (2011)
11. Ralphs, T.K., Galati, M.V.: DIP – decomposition for integer programming (2009), https://projects.coin-or.org/Dip
12. Sherali, H.D., Lee, Y., Kim, Y.: Partial convexification cuts for 0-1 mixed-integer programs. European J. Oper. Res. 165(3), 625–648 (2005)
13. Tebboth, J.R.: A Computational Study of Dantzig-Wolfe Decomposition. PhD thesis, University of Buckingham (2001)
14. Vanderbeck, F.: BaPCod – a generic branch-and-price code (2005), https://wiki.bordeaux.inria.fr/realopt/pmwiki.php/Project/BaPCod
15. Vanderbeck, F., Wolsey, L.: Reformulation and decomposition of integer programs. In: Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds.) 50 Years of Integer Programming 1958–2008. Springer, Berlin (2010)

Lift-and-Project Cuts for Mixed Integer Convex Programs

Pierre Bonami

LIF, CNRS Aix-Marseille Université, 163 avenue de Luminy - Case 901 F-13288
Marseille Cedex 9 France
pierre.bonami@lif.univ-mrs.fr

Abstract. This paper addresses the problem of generating cuts for mixed integer nonlinear programs where the objective is linear and the relations between the decision variables are described by convex functions defining a convex feasible region. We propose a new method for strengthening the continuous relaxations of such problems using cutting planes. Our method can be seen as a practical implementation of the lift-and-project technique in the nonlinear case. To derive each cut we use a combination of a nonlinear programming subproblem and a linear outer approximation. One of the main features of the approach is that the subproblems solved to generate cuts are typically not more complicated than the original continuous relaxation. In particular they do not require the introduction of additional variables or nonlinearities. We propose several strategies for using the technique and present preliminary computational evidence of its practical interest. In particular, the cuts allow us to improve over the state of the art branch-and-bound of the solver Bonmin, solving more problems in faster computing times on average.

Keywords: Mixed Integer Nonlinear Programming, Disjunctive Programming, Lift-and-Project, Cutting Planes.

1 Introduction

In this paper we consider mixed integer nonlinear programs of the form

$$\begin{array}{ll} \min\ c^T x & \\ g_i(x) \leq 0 & i = 1, \ldots, m \\ x_j \in \mathbb{Z} & j = 1, \ldots, p \\ x_j \in \mathbb{R} & j = p+1, \ldots, n \end{array} \tag{MICP}$$

where $1 \leq p \leq n$, $c \in \mathbb{R}^n$ and for $i = 1, \ldots, m$, $g_i : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a continuously differentiable convex function. Because of the convexity of the function g_i we call this problem a mixed integer convex program. Note that our problem formulation does not have bounds on the variables. If any are present, we assume that they are among the constraints $g_i(x) \leq 0$. Note also that any

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 52–64, 2011.

mixed-integer minimization problem with a convex objective function and lower or equal constraints described by convex functions can be put into the form of (MICP) by adding a variable and putting the objective in the constraints.

MICP can be seen as a generalization of Mixed Integer Linear Programming. It has received a sustained level of attention in recent years. In particular, a successful line of research has been to try and extend techniques that have been succesfull in the solution of MILPs to MICP. This has led to the design of algorithms and solvers that can effectively solve MICPs of relatively large size [1,12] (see also [14] for a recent survey of these efforts). One aspect that is still missing in that context is efficient techniques for strengthening the continuous relaxation of (MICP).

Cutting plane techniques are one of the main ingredients used to achieve that in modern mixed-integer linear programming solvers. Among the most widely used cuts are Gomory's Mixed Integer cuts [22,6], Mixed Integer Rounding cuts [27], Extended covers [10]...

Inspired by these methods, several cutting plane approaches have been proposed for MICP. In particular, Ceria, Soares [17] and Stubbs, Mehrotra [31] have proposed generalizations of disjunctive programming [3] and the lift-and-project approach [5] to nonlinear convex sets and MICP. Unfortunately these approaches require the solution of nonlinear programs in extended spaces and which feature so called perspective functions that are not handled well by general purpose nonlinear programming algorithms (these functions are not differentiable everywhere). Probably because of these difficulties, to the best of our knowledge, these methods have never been tested on a large scale and are not included in solvers. Some other approaches have been developed for specific classes of MICPs. In particular, several methods have been proposed for MICPs where the nonlinear constraints are conic constraints. Cezik and Iyengar [18] have generalized the Chvátal-Gomory and lift-and-project cuts. Atamtürk and Narayanan [2] have proposed a generalization of MIR. These approaches have given encouraging computational results but they are restricted to specific types of problems.

Our goal in this paper is to propose a cutting plane technique for MICP that can be applied to the general case and that is computationally good enough to be used in solvers as an effective tool for strengthening continuous relaxations. Our approach is closely related to lift-and-project techniques in MILP, in particular it can be seen as a generalization of [9,11] to MICP.

Suppose that we are given to cut a point $\overline{x} \in \mathbb{R}^n$ that does not satisfy the integrality requirements of (MICP). Similarly to what is done in MILP, our approach tries to generate one cut for each variable x_j, $1 \leq j \leq p$, such that $\overline{x}_j \notin \mathbb{Z}$. For each x_j, the cut is built in two phases. In the first phase, we solve a nonlinear program (NLP for short) that tells us if a cut that separates $\overline{x}$ by using the integrality of x_j exists. If the answer to the first phase is yes, the second phase builds an outer approximation of (MICP) and derives a cut from it by Linear Programming (LP). An important property is that a cut is always generated in the second phase if the answer to the first phase is yes and a constraint qualification holds. Another desirable property is that the nonlinear program to

solve in the first phase is defined in the same space as the original problem and is typically not more difficult to solve than the continuous relaxation of (MICP). We propose two different strategies to use this cutting plane technique and test them within the solver Bonmin [12,13]. Our conclusions are positive in that we are able to close a significant proportion of the integrality gap on several classes of problems and this helps to effectively solve some hard problems faster with branch-and-bound.

Let us note that very recently, another proposal for a practical cutting plane approach for (MICP) has been made by Linderoth, Kılınç and Luedtke [23]. This approach and the one presented here bear some similarities but also have some major differences. The main one, is that the approach in [23] does not use nonlinear programming but solely relies on an outer approximation of (MICP) refined by constraint generation. The advantage of [23] is that it may be more lightweight in some cases but the disadvantage is that the constraint generation can have very slow convergence properties. We benefit from the good convergence properties offered by sophisticated nonlinear programming algorithms.

The paper is organized as follows. In Sect. 2, we outline our procedure for generating each cut. In Sect. 3, we present in details the NLP used in the first phase of our algorithm and establish some of its main properties. In Sect. 4, we present the LP used in the second phase of our algorithm. In Sect. 5, we describe two strategies for using our cutting planes. Finally, in Sect. 6, we report on the experimental testing of our method.

2 Outline of the Approach

We denote by $C := \{x \in \mathbb{R}^n : g_i(x) \leq 0,\ i = 1, \ldots, m\}$ the feasible set of the continuous relaxation of (MICP) and by $X := \operatorname{conv}\left(C \cap \left(\mathbb{Z}^p \times \mathbb{R}^{n-p}\right)\right)$ the convex hull of the feasible solutions of (MICP). Let $\overline{x} \in C \setminus \left(\mathbb{Z}^p \times \mathbb{R}^{n-p}\right)$ be the fractional point that we want to separate from X. Note that we do not make further assumptions on $\overline{x}$. In particular, we do not assume that it is an extreme or an exposed point of C and it may even belong to X (of course in such a case our approach would not find any cut).

To generate inequalities, we use a *split relaxation* of X which is defined as follows. We call split an integral vector $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$ such that $\pi_j = 0$, $j = p+1, \ldots, n$. We then define the split relaxation corresponding to (π, π_0) as

$$C^{(\pi,\pi_0)} := \operatorname{conv}\left(C \cap \left(\left\{x \in \mathbb{R}^n : \pi^T x \leq \pi_0\right\} \cup \left\{x \in \mathbb{R}^n : \pi^T x \geq \pi_0 + 1\right\}\right)\right).$$

Clearly $X \subseteq C^{(\pi,\pi_0)}$ since, for any $x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$, $\pi^T x \in \mathbb{Z}$ and $\mathbb{Z} \cap]\pi_0, \pi_0 + 1[= \emptyset$.

Given a split (π, π_0), our procedure decides if $\overline{x} \in C^{(\pi,\pi_0)}$ and finds a separating hyperplane if this not the case. In this paper, we only use elementary splits of the form $(e_k, \lfloor \overline{x}_k \rfloor)$ (where $1 \leq k \leq p$ and e_k is the k-th unit vector). Nevertheless our method applies to any split. In this section we give a geometric outline of the approach using a general split.

Our approach is based on the following simple lemma that states a necessary and sufficient condition for $\overline{x}$ to belong to $C^{(\pi,\pi_0)}$.

Lemma 1. *Let $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$ be a valid split and let $\overline{x} \in C$ be such that $\pi^T\overline{x} - \pi_0 \in]0,1[$. $\overline{x} \in C^{(\pi,\pi_0)}$ if and only if there exists $x^1 \in C$ such that*

(i) $\frac{\overline{x}-\left(\pi^T\overline{x}-\pi_0\right)x^1}{1-\pi^T\overline{x}+\pi_0} \in C$,
(ii) $\pi^T x^1 \geq \pi_0 + 1$.

Proof. Suppose first that $\overline{x} \in C^{(\pi,\pi_0)}$, then there exists $x^0, x^1 \in C$ and $\lambda \in [0,1]$ such that $\pi^T x^0 \leq \pi_0$, $\pi^T x^1 \geq \pi_0 + 1$ and $\overline{x} = \lambda x^1 + (1-\lambda)x^0$. Note that, without loss of generality, we can assume that $\pi^T x^0 = \pi_0$ and $\pi^T x^1 = \pi_0 + 1$ (just take the intersections of the initial segment $[x^0, x^1]$ with the two hyperplanes $\pi^T x = \pi_0$ and $\pi^T x = \pi_0 + 1$). Condition (ii) clearly holds. Also, since $\pi^T(x^1 - x^0) = 1$, $\pi^T\overline{x} = \lambda\pi^T x^1 + (1-\lambda)\pi^T x^0 = \lambda + \pi^T x^0$, and $\lambda = \pi^T\overline{x} - \pi_0$. Condition (i) follows.

Suppose now that $x^1 \in C$ satisfies (i) and (ii). Let $x^0 := \frac{\overline{x}-\left(\pi^T\overline{x}-\pi_0\right)x^1}{1-\pi^T\overline{x}+\pi_0}$. By (i), $x^0 \in C$. Furthermore, we clearly have $\overline{x} = \left(\pi^T\overline{x} - \pi_0\right)x^1 + \left(1 - \pi^T\overline{x} + \pi_0\right)x^0$. Since $\pi^T x^1 \geq \pi_0 + 1$ by hypothesis, we just need to verify that $\pi^T x^0 \leq \pi_0$ to prove the result. Indeed, $\pi^T x^0 = \frac{\pi^T\overline{x}-\left(\pi^T\overline{x}-\pi_0\right)\pi^T x^1}{1-\pi^T\overline{x}+\pi_0} \leq \frac{\pi^T\overline{x}-\left(\pi^T\overline{x}-\pi_0\right)(\pi_0+1)}{1-\pi^T\overline{x}+\pi_0} = \pi_0\frac{1-\pi^T\overline{x}+\pi_0}{1-\pi^T\overline{x}+\pi_0}$. □

We are now ready to describe the two phases of our separation procedure for $C^{(\pi,\pi_0)}$. In the first phase, we find $\overline{x}^1 \in C$, satisfying condition (i) of the lemma and maximizing $\pi^T x$. In Sect. 3, it will be shown that this can be accomplished by solving a convex NLP formulated in $\mathbb{R}^n$. If $\pi^T\overline{x}^1 \geq \pi^0 + 1$, $\overline{x}^1$ proves that $\overline{x} \in C^{(\pi,\pi_0)}$ and we can not generate a cut using this split. Now suppose that $\pi^T\overline{x}^1 < \pi^0 + 1$. Then $\overline{x} \notin C^{(\pi,\pi_0)}$ and we enter the second phase of the procedure. We build an outer approximation $P(\overline{x}^0, \overline{x}^1)$ of C by taking the first order approximations of the constraints of C in the points $\overline{x}^1$ and $\overline{x}^0 := \frac{\overline{x}-\left(\pi^T\overline{x}-\pi_0\right)\overline{x}^1}{1-\pi^T\overline{x}+\pi_0}$:

$$P(\overline{x}^0,\overline{x}^1) = \big\{x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} : \nabla g_i(\overline{x}^1)^T\left(x - \overline{x}^1\right) + g_i(\overline{x}^1) \leq 0,\ i = 1,\ldots,m,$$
$$\nabla g_i(\overline{x}^0)^T\left(x - \overline{x}^0\right) + g_i(\overline{x}^0) \leq 0,\ i = 1,\ldots,m\big\}.$$

Similarly to $C^{(\pi,\pi_0)}$, we define the split relaxation $P(\overline{x}^0,\overline{x}^1)^{(\pi,\pi_0)}$ of $P(\overline{x}^0,\overline{x}^1)$. A cut separating $\overline{x}$ from $P(\overline{x}^0,\overline{x}^1)^{(\pi,\pi_0)}$ can then be sought using classical polyhedral disjunctive programming techniques. We will show in Sect. 4 that, such a cut can typically be found by solving an LP in $\mathbb{R}^n$. In particular, we show that, if the optimal solution of the NLP solved in the first phase satisfies a constraint qualification, a cut is always found by solving this LP.

The fact that the solution of the NLP solved in the first phase has to satisfy a constraint qualification for our procedure to generate a cut in the second phase is not to be overlooked. In particular, we will show that whenever $\overline{x}$ is an extreme point of C the NLP for finding $\overline{x}^1$ as defined above leads to a degenerate NLP where no constraint qualification holds. The procedure will be slightly amended to solve this issue.

3 Phase I: Testing If $\overline{x}$ Belongs to a Split Relaxation

In this section, we derive the NLP solved in the first phase of our procedure to decide if $\overline{x} \in C^{(\pi,\pi_0)}$. Since in practice here, we only separate cuts using elementary splits of the form $(e_k, \lfloor \overline{x}_k \rfloor)$ (where $1 \leq k \leq p$), from now on, for simplicity's sake, we restrict ourselves to this case (the procedure remains valid for any split). We denote the fractional part of $\overline{x}_k$ by $f_0 := \overline{x}_k - \lfloor \overline{x}_k \rfloor$ and assume $f_0 > 0$.

Consider the optimization problem

$$\begin{aligned} \max\ & y_k - f_0 \lceil \overline{x}_k \rceil \\ & g_i\left(\frac{y}{f_0}\right) \leq 0 \qquad i = 1, \ldots, m, \\ & g_i\left(\frac{\overline{x} - y}{1 - f_0}\right) \leq 0 \quad i = 1, \ldots, m. \end{aligned} \tag{MNLP}$$

Let y is be an optimal solution to (MNLP), $x^1 := \frac{y}{f_0}$ satisfies $x^1 \in C$ and condition (i) of Lemma 1. If furthermore, the objective value is non-negative, x^1 satisfies condition (ii) of the lemma and therefore $\overline{x} \in C^{(e_k, \lfloor \overline{x}_k \rfloor)}$. Therefore, solving (MNLP) solves the phase I of our procedure as defined in Sect. 2.

Note that (MNLP) is a convex program: by definition, all the functions g_i are convex and the nonlinear functions describing the constraints of (MNLP) are obtained by composing g_i with an affine function (either $\frac{y}{f_0}$ or $\frac{\overline{x}-y}{1-f_0}$). (MNLP) has the same number of variables as our original problems and twice as many constraints. Note that if there are any linear or bound constraints in the original problem (MNLP) can be made smaller in practice. Indeed if g_i is an affine function, $g_i(\frac{y}{f_0}) \leq 0$ and $g_i(\frac{\overline{x}-y}{1-f_0}) \leq 0$ can be reformulated as $g_i(\overline{x}) + (1 - f_0)g_i(0) \leq g_i(y) \leq (1 - f_0)g_i(0)$. Therefore, (MNLP) can be reformulated as a problem that has as many linear constraints as (MICP) and twice as many nonlinear constraints.

We note that (MNLP) is a generalization in the nonlinear setting of a separation problem recently proposed [20,11]. In that respect, (MNLP) is related to the famous Cut Generation LP used in disjunctive programming [5] in that, in the linear case, the dual of (MNLP) is equivalent to the standard CGLP of lift-and-project procedures with a particular normalization condition (see [11] for the precise equivalence). A well-known property of the linear version of (MNLP) is that, if $\overline{x}$ is an extreme point of the continuous relaxation, the solution of (MNLP) is unique (and actually corresponds to the GMI cut) [19,11]. Next lemma generalizes this result to our nonlinear setting.

Lemma 2. *Let $\overline{x} \in C$, be such that $\overline{x}_k \notin \mathbb{Z}$. If $\overline{x}$ is an extreme point of C, then the unique solution to (MNLP) is $y = f_0\overline{x}$.*

Proof. It follows directly from the fact that if $\overline{x}$ is an extreme point, then by definition, the only $x^1 \in C$ satisfying condition (i) of Lemma 1 is $\overline{x}$ itself. □

Lemma 2 could be seen as positive. Indeed, if $\overline{x}$ is an extreme point of C then (MNLP) can be solved by a closed form formula and furthermore (MNLP) shows that $\overline{x} \notin C^{(e_k, \lfloor \overline{x}_k \rfloor)}$ since $f_0(\overline{x}_k - \lfloor \overline{x}_k \rfloor) < 0$. Lemma 2 actually has rather dire implications. Indeed, if $y = f_0\overline{x}$, then, for $i = 1, \ldots, m$, the two nonlinear inequalities $g_i(\frac{y}{f_0}) \leq 0$ and $g_i(\frac{\overline{x}-y}{1-f_0}) \leq 0$ are identical and therefore no constraint qualification holds in y. It follows, that no usefull dual information exists. Without dual information, it is hard to imagine how a cut can be derived. Note that, in general, the optimal solution of the continuous relaxation of (MICP) should be expected to be an extreme point (this is at least true when the optimum is unique). Therefore it should be expected that the point $\overline{x}$ we want to cut is an extreme point of C. Clearly, this issue needs to be addressed if we want to be able to strengthen the continuous relaxation in a systematic way.

To resolve this issue, we make a small modification to (MNLP). Consider the following nonlinear program:

$$\begin{aligned} \max\ & y_k - f_0\lceil \overline{x}_k \rceil \\ & g_i\left(\frac{y}{f_0}\right) \leq f_0\lceil \overline{x}_k \rceil - y_k \qquad i = 1, \ldots, m, \\ & g_i\left(\frac{\overline{x}-y}{1-f_0}\right) \leq f_0\lceil \overline{x}_k \rceil - y_k \quad i = 1, \ldots, m. \end{aligned} \tag{MNLP'}$$

The difference between (MNLP') and (MNLP) is the term $f_0\lceil \overline{x}_k \rceil - y_k$ that we add to the right-hand-side of the constraints. The effect of this term is to relax the constraints whenever $y_k - f_0\lceil \overline{x}_k \rceil < 0$ (therefore in particular when $y = f_0\overline{x}$) while keeping the property that if the optimal solution to (MNLP') is non-negative $x^1 := \frac{y}{f_0}$ satisfies the conditions of Lemma 1. We can not claim that a constraint qualification always holds at the optimum of (MNLP') but our practical experiment is that it is typically not an issue.

The modification made from (MNLP) to (MNLP') may seem puzzling. It is actually connected to a well known technique of lift-and-project. Indeed, if we again assume that all functions g_i are linear, it can be shown that (MNLP') is equivalent to the CGLP with the normalization condition $\sum(u_i + v_i) + u_0 + v_0 = 1$ first presented in [8]. Therefore, the modification made from (MNLP) to (MNLP') can be seen as a change in the normalization condition from $u_0 + v_0 = 1$ to $\sum(u_i + v_i) + u_0 + v_0 = 1$. In this light, the change can be seen as even more desirable since in the context of MILP several experiments have concluded that the latter typically gives better cuts [29,4,19]. Note that in practice, in this paper, we only add the term $f_0\lceil \overline{x}_k \rceil - y_k$ to the nonlinear constraints of (MNLP). Doing this is enough to avoid the numerical difficulties and allows to keep (MNLP') compact (if the term is added to linear constraints then their number has to be doubled).

4 Phase II: Computing the Cut

We now turn to the description of the second phase of the procedure. At the beginning of Phase II, we have a point $\overline{y}$ optimal solution of (MNLP') and

such that $\frac{\overline{y}}{f_0} - \lceil \overline{x}_k \rceil < 0$. Note that, because (MNLP') relaxed the constraints of (MNLP), $\frac{\overline{y}}{f_0} \notin C$ and condition (i) of Lemma 1 is not satisfied. But the maximality of $\overline{y}$ still proves that $\overline{x} \notin C^{(\pi,\pi_0)}$.

As explained in Sect. 2, we now build an outer approximation $P(\overline{x}^0, \overline{x}^1)$ of C by taking linear approximations in $\overline{x}^1 := \frac{\overline{y}}{f_0}$ and $\overline{x}^0 := \frac{\overline{x}-\overline{y}}{1-f_0}$ (note that the validity of this outer approximation does not depend on the fact that $\overline{x}^1$ and $\overline{x}^0 \in C$).

Once this outer-approximation is built, a cut can be found by classical disjunctive programming techniques. In our case, we chose to solve an LP which is just the linear version of (MNLP) formulated for $P(\overline{x}^0, \overline{x}^1)^{(e_k, \lfloor \overline{x}_k \rfloor)}$:

$$
\begin{aligned}
&\max\ y_k - f_0 \lceil \overline{x}_k \rceil \\
&\nabla g_i(\overline{x}^1)^T y \geq \nabla g_i(\overline{x}^1)^T \overline{x} + (1 - f_0)\left(g_i(\overline{x}^1) - \nabla g_i(\overline{x}^1)^T \overline{x}^1\right) && i = 1, \ldots, m, \\
&\nabla g_i(\overline{x}^1)^T y \leq -f_0\left(g_i(\overline{x}^1) - \nabla g_i(\overline{x}^1)^T \overline{x}^1\right), && i = 1, \ldots, m, \\
&\nabla g_i(\overline{x}^0)^T y \geq \nabla g_i(\overline{x}^0)^T \overline{x} + (1 - f_0)\left(g_i(\overline{x}^0) - \nabla g_i(\overline{x}^0)^T \overline{x}^0\right), && i = 1, \ldots, m, \\
&\nabla g_i(\overline{x}^0)^T y \leq -f_0\left(g_i(\overline{x}^0) - \nabla g_i(\overline{x}^0)^T \overline{x}^0\right), && i = 1, \ldots, m, \\
&x \in \mathbb{R}^n.
\end{aligned}
\tag{MLP($\overline{x}^0, \overline{x}^1$))}
$$

It follows directly from linear programming duality that a cut can be constructed from any dual feasible basis of (MLP($\overline{x}^0, \overline{x}^1$)) with negative primal cost (see for example [11] for details).

Note that other Cut Generation LPs could be used to find a cut separating $\overline{x}$ from $P(\overline{x}^0, \overline{x}^1)^{(e_k, \lfloor \overline{x}_k \rfloor)}$ in this step. One advantage of (MLP($\overline{x}^0, \overline{x}^1$)) is its size. Indeed, it does not require the introduction of extra variables and can typically be solved very quickly. Note also, that the cut obtained from the solution of (MLP($\overline{x}^0, \overline{x}^1$)) can be strengthened by the classical monoidal strengthening method [7,5].

As the next theorem shows, a fundamental property of (MLP($\overline{x}^0, \overline{x}^1$)) is that if a constraint qualification holds at the optimum of (MNLP') then the optimal value of (MLP($\overline{x}^0, \overline{x}^1$)) is lower or equal to that of (MNLP') (and therefore a cut can be generated).

Theorem 1. *Let $\overline{y}$ be an optimal solution of (MNLP') satisfying a constraint qualification. Let $\overline{x}^1 := \frac{\overline{y}}{f_0}$ and $\overline{x}^0 := \frac{\overline{x}-\overline{y}}{1-f_0}$, and let $\hat{y}$ be a solution to (MLP($\overline{x}^0, \overline{x}^1$)). If $\overline{y}_k < f_0 \lceil \overline{x}_k \rceil$, then $\hat{y}_k \leq \overline{y}_k$*

The proof follows directly from the KKT conditions and the definition of (MLP($\overline{x}^0, \overline{x}^1$)).

This theorem completes the description of our algorithm for finding a cut for $C^{(e_k, \lfloor \overline{x}_k \rfloor)}$. It is summarized in Algorithm 1.

5 Cutting Plane Strategies and Practical Considerations

In the next section, we present preliminary computational testing of Algorithm 1. First, we discuss here strategies for using it. Note that we did not specify yet

Algorithm 1. Cut Generation Algorithm

Input. A convex set C, $\overline{x} \in C$ such that $\overline{x}_k \notin \mathbb{Z}$, and a split $(e_k, \lfloor \overline{x}_k \rfloor)$.

I. **Testing if $\overline{x} \in C^{(e_k, \lfloor \overline{x}_k \rfloor)}$.**
Solve (MNLP'), if the objective value is non-negative **STOP**, otherwise let $\overline{y}$ be the optimal solution and **go to Step II**.

II. **Computing the cut.**
Let $\overline{x}^1 := \frac{\overline{y}}{f_0}$ and $\overline{x}^0 = \frac{\overline{x}-\overline{y}}{1-f_0}$. Solve (MLP($\overline{x}^0, \overline{x}^1$)), build a cut $\alpha^T x \geq \beta$ from the dual multipliers and strengthen it. **Return the strengthened cut** $\alpha^T x \geq \beta$.

how to choose the point to cut $\overline{x}$ and the elementary disjunctions. We test two strategies that were previously proposed and extensively used in the context of MILP. We also discuss here the problem of cut validity.

5.1 Separation by Rounds

Our first strategy consists in separating cuts by rounds following the approach first proposed in [5]. Let $\overline{x}$ be the optimal solution of the continuous relaxation of (MICP). If $\overline{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$ the problem is solved and we stop. Otherwise, a round of cuts consists in applying Algorithm 1 for each $k \in \{1, \ldots, p\}$ such that $\overline{x}_k \notin \mathbb{Z}$. At the end of the round all the cuts found are added to the formulation of (MICP). The process is then iterated recursively with the strengthened formulation of (MICP).

It is important to note that in this approach, the sub-problems (MNLP') and (MLP($\overline{x}^0, \overline{x}^1$)) are augmented in each new round with the cuts found in previous rounds. This approach has desirable properties: by Lemma 2 and Theorem 1 if $\overline{x}$ is an extreme point of C, it is guaranteed that cuts will be found. As noted in the context of MILP it can also have undesirable consequences. First increasing the rank of the cuts at each iteration may lead to numerical difficulties and invalid cuts. Second increasing the sizes of (MNLP') and (MLP($\overline{x}^0, \overline{x}^1$)) can make them very difficult to solve. In practice, we only do a very limited number of rounds. We test this approach with 1 and 10 rounds of cuts.

5.2 Rank-1 Optimization

In this other strategy, we limit the rank of the cut we generate to 1. Essentially, this means that we apply the same procedure as before except that the cuts found are never added to (MNLP') and (MLP($\overline{x}^0, \overline{x}^1$)). Of course, this means that contrary to before the procedure may stop with $\overline{x} \notin \mathbb{Z}^p \times \mathbb{R}^{n-p}$ because no rank-1 cut exists anymore. Nevertheless, it has been experimentally observed in the context of MILP that this strategy sometimes allows to close more gap faster [15,11]. It also gives a feeling of the strength of the closure obtained by using only rank 1 cuts.

An aspect that has to be dealt with in this strategy is to try to avoid solving too many (MNLP') that do not generate cuts. To deal with this, we follow the strategies proposed in [11].

Finally, although this algorithm may eventually stop because no rank-1 cut can be generated anymore, it might take a very large amount of time and iterations. Here, we limit the computations to 10 minutes of CPU time and 100 rounds.

5.3 Cut Validity and Stability

The problem of generating inequalities that cuts off integer feasible solutions is a well known issue in MILP. It should be expected to be even more acute when some constraints are nonlinear. Here, we make a few observations on our experience of dealing with this issue during the experiments presented in the next section. First, let us state that we can not guarantee that all the cuts that we generate are valid. The only thing that we can say is that the integer optimal solutions were never cut during our experiments.

We actually do not believe our approach to be more prone to generating invalid cuts than other algorithms for (MICP). Note that the cuts that we generate are solely computed through a linear outer approximation of (MICP). In particular they do not depend on the precision of the solution of (MNLP'). As long as the outer approximation $P(\overline{x}^0, \overline{x}^1)$ is built with care (in practice, as is customary, we try to avoid difficulties there by slightly relaxing each constraint), the approach is not more dangerous than any other Outer Approximation based algorithm.

This does not mean that we did not experience troubles. Most of the difficulties we experienced came from numerical stability issues when solving the continuous relaxation of (MICP) augmented with many cuts. We used the solver filterSQP [21] to solve this NLP and experienced that, as it grows larger, it can rapidly become unsolvable (note that filterSQP is an active set method, it is well known that such methods may not be best suited when problems sizes grow). To avoid those difficulties, we had to take drastic measures in rejecting some cuts based on their numerical properties. A criterion that seemed to play a particularly important role is the ratio between the largest and smallest absolute values of the non-zero coefficients of a cut. It is usual in MILP to accept cuts until this ratio is as high as 10^8 without experiencing much difficulties (see for example [28]). Here, to stand clear of numerical troubles, we had to reject all cuts for which this ratio was greater than 10^4. In the rank 1 experiments, we also had to clean (MICP) from any inactive cuts regularly (those cuts were kept in a pool in case they would be needed later on).

6 Computational Testing

Algorithm 1 and the two strategies outlined in Sect. 5.1 and 5.2 were implemented in C++ using the interfaces of the Bonmin [13,12] framework from COIN-OR [26]. In particular, our code allows to use any LP solver with an OSI [26] interface to solve (MLP$(\overline{x}^0, \overline{x}^1)$) and any NLP solver with a TNLP interface [32] to solve (MNLP'). Here we used CPLEX 12.1 and FilterSQP respectively. All experiments were conducted on a machine equipped with Intel Quad Core Xeon

2.93GHz processors and 120 GiB of RAM, using only one thread for each run. The test set consists of instances collected from different sources [24,16,30,25] (see within [14] for references of each instances types). Among about 150 instances collected, we selected all instances that took more than 1000 nodes to solve with a basic branch-and-bound algorithm. We then removed all instances that had a nonlinear objective and all instances that could not be solved in three hours with any of the methods tested here. As a result we obtain a test set of 80 instances. As noted in the introduction, instances with a nonlinear objective can be put into the form of (MICP) and therefore treated in a systematical way by our approach. Nevertheless, we chose to exclude them because we are unsure if this is the correct treatment for nonlinear objectives (it posed some stability issues in practice) and we felt there were enough interesting instances to report without them.

6.1 Gap Closed

In this first experiment we used the various cut generation strategies to only strengthen the continuous relaxations. The three strategies tested were 1 round of cuts, 10 rounds of cuts, and the rank 1 optimization. Our goal was to measure the CPU time of each method and the percentage of the integrality gap it closes[1]. The results are summarized in Table 1.

Table 1. Number of instances in each class and then for each method and each instances class: averages cut generation times in seconds, average number of cuts and percentage of gap closed

		1 round			10 rounds			rank 1		
Type	#	CPU	# cuts	gap	CPU	# cuts	gap	CPU	# cuts	gap
`Batch`	4	2.85	24.25	20.64	40.65	89.5	24.01	60.08	112.75	24.06
`CLay`	7	1.64	40.14	1.55	16.16	219.71	8.55	39.71	152.29	33.71
`FLay`	5	0.35	29.75	1.25	7.74	225.75	3.85	48.19	400.5	34.98
`RSyn`	16	9.15	64.63	21.32	107.77	406.13	47.61	247.81	584.94	59.96
`SLay`	9	12.92	82.33	4.60	182.76	254.56	7.58	230.25	207.44	29.65
`Syn`	18	3.78	58.22	23.58	92.03	357.67	56.36	110.91	622.67	84.52
`fo-m`	5	0.27	32.4	0.00	7.93	139.8	0.00	12.79	283.2	0.00
`nd`	3	2.31	22.33	6.32	33.95	52.67	11.05	251.79	312.67	69.16
`sssd`	12	0.12	24.92	3.75	3.42	219.83	43.12	1.85	141.67	97.97
`trimloss`	1	0.27	11	2.70	3.35	106	12.15	10.02	96	12.32
all instances	80	4.57	48.99	12.18	69.07	276.01	32.30	120.08	377.24	58.05

The results shows that our various methods are able to close significant portions of the integrality gap in reasonable computing times. In particular, on average on our test set, the rank 1 approach closes 58% of the gap in about two minutes.

[1] If z_C is the optimal value of the initial continuous relaxation of (MICP), z_X is the optimal value of (MICP) and z_S is the optimal value of the continuous relaxation strengthened by cutting planes, the percentage of integrality gap closed is $100\left(1-\frac{z_X-z_S}{z_X-z_C}\right)$.

6.2 Complete Resolutions

In this second experiment we want to assess if the cuts can be useful to effectively solve the test instances to optimality faster. To this end, after the cut generation procedure is finished, we keep in the formulation all cuts that are tight at the new continuous optimum and solve the formulation with Bonmin's NLP branch-and-bound algorithm `B-BB`. We then compare the solution process with `B-BB` without cuts. The results are reported in Table 2.

Table 2. For each method and each instances type: number of instances solved (#I), average total solution times (include cut generation) and branch-and-bound nodes (averages figures taken only on the subsets of instances solved by all methods)

	no cuts			1 round			10 rounds			rank-1		
Type	#I	CPU	nodes	#I	CPU	nodes	#I	CPU	nodes	#I	CPU	nodes
`Batch`	4	101.6	1681	4	75.2	1069	4	127.6	1316	4	137.2	1110
`Clay`	7	106.3	13435	7	117.1	10727	7	171.5	11616	7	199.0	10875
`Flay`	5	1175.5	51773	4	1190.7	48522	4	1260.8	49964	4	1979.3	55611
`Rsyn`	10	2689.5	38914	11	2378.8	19422	12	1535.0	6082	16	1368.1	988
`Slay`	9	152.2	1455	9	177.4	1395	9	329.8	1274	9	392.5	1055
`Syn`	12	546.9	21476	12	710.2	21534	13	310.4	5309	18	94.5	54
`fo-m`	4	2271.2	315700	4	2019.4	278629	4	2017.5	209537	5	2764.0	263231
`nd`	3	1397.3	7946	3	1436.0	7763	3	1427.1	6821	3	524.5	527
`sssd`	8	362.8	69713	8	115.9	19207	6	200.4	34544	10	207.7	36877
`tls`	1	2071.1	1298667	1	1998.5	1092360	1	3636.2	1336885	1	4889.5	1517275
solved	63	980	66130	63	928	51946	63	776	47299	77	792	52190

The results show that the cuts are indeed helping to solve more instances faster. The rank-1 procedure can solve 77 instances, when all others solve only 63. On instances solved by all approaches, rank-1 and 10 rounds are the two most competitive. It should be noted that the usefulness of the cuts varies considerably between different types of instances. Cuts are particularly usefull for solving the `RSyn`, `Syn`, `nd` and `sssd` instances. On the other hand they seem to only slow down the solution of `CLay`, `FLay`, `SLay` and `trimloss`.

Acknowledgments. Research supported by ANR grant ANR06-BLAN-0375 and by a Google Focused Research Award.

References

1. Abhishek, K., Leyffer, S., Linderoth, J.: FilMINT: An Outer Approximation-Based Solver for Convex Mixed-Integer Nonlinear Programs. INFORMS Journal on Computing 22, 555–567 (2010)
2. Atamtürk, A., Narayanan, V.: Conic mixed integer rounding cuts. Mathematical Programming 122, 1–20 (2010)
3. Balas, E.: Disjunctive programming: Properties of the convex hull of feasible points. Discrete Applied Mathematics 89, 3–44 (1988); (originaly MSRR # 348, Carnegie Mellon University, July 1974)

4. Balas, E., Bonami, P.: Generating lift-and-project cuts from the LP simplex tableau: open source implementation and testing of new variants. Mathematical Programming Computations 1, 165–199 (2009)
5. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. Math. Programming 58, 295–324 (1993)
6. Balas, E., Ceria, S., Cornuéjols, G., Natraj, N.: Gomory cuts revisited. Operations Research Letters 19, 1–9 (1996)
7. Balas, E., Jeroslow, R.G.: Strengthening cuts for mixed integer programs. European J. Oper. Res. 4(4), 224–234 (1980)
8. Balas, E., Perregaard, M.: Lift and project for mixed 0-1 programming: Recent progress. Discrete Applied Mathematics 123(1-3), 129–154 (2002)
9. Balas, E., Perregaard, M.: A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming. Math. Program 94(2-3, Ser. B), 221–245 (2003); The Aussois 2000 Workshop in Combinatorial Optimization
10. Balas, E., Zemel, E.: Facets of the knapsack polytope from minimal covers. SIAM Journal on Applied Mathematics 34, 119–148 (1978)
11. Bonami, P.: On optimizing over lift-and-project closures. Research Report HAL, CNRS (October 2010), http://hal.archives-ouvertes.fr/hal-00529816/en/
12. Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A.: An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optimization 5(2), 186–204 (2008)
13. Bonami, P., Forrest, J.J.H., Laird, C., Lee, J., Margot, F., Wächter, A.: Bonmin: Basic Open-source Nonlinear Mixed INteger programming (July 2006), http://www.coin-or.org/Bonmin
14. Bonami, P., Kılınç, M., Linderoth, J.: Algorithms and Software for Convex Mixed Integer Nonlinear Programs. Technical report, Technical Report #1664, Computer Sciences Department, University of Wisconsin-Madison (2009)
15. Bonami, P., Minoux, M.: Using rank-1 lift-and-project closures to generate cuts for 0–1 MIPs, a computational investigation. Discrete Optimization 2(4), 288–307 (2005)
16. Bussieck, M.R., Drud, A.S., Meeraus, A.: MINLPLib – A collection of test models for mixed-integer nonlinear programming. INFORMS Journal on Computing 15(1) (2003)
17. Ceria, S., Soares, J.: Convex programming for disjunctive optimization. Mathematical Programming 86, 595–614 (1999)
18. Cezik, M.T., Iyengar, G.: Cuts for mixed 0-1 conic programming. Mathematical Programming 104, 179–202 (2005)
19. Fischetti, M., Lodi, A., Tramontani, A.: On the separation of disjunctive cuts. Mathematical Programming (2009), doi: 10.1007/s10107-009-0300-y (in press)
20. Fischetti, M., Salvagnin, D.: An in-out approach to disjunctive optimization. In: Lodi, A., Milano, M., Toth, P. (eds.) CPAIOR 2010. LNCS, vol. 6140, pp. 136–140. Springer, Heidelberg (2010)
21. Fletcher, R., Leyffer, S.: User manual for filterSQP, University of Dundee Numerical Analysis Report NA-181 (1998)
22. Gomory, R.E.: An algorithm for integer solution solutions to linear programming. In: Graves, R.L., Wolfe, P. (eds.) Recent Advances in Mathematical Programming, pp. 269–302. McGraw-Hill, New York (1963)

23. Kılınç, M., Linderoth, J., Luedtke, J.: Effective separation of disjunctive cuts for convex mixed integer nonlinear programs. Technical Report Computer Sciences Department, University of Wisconsin-Madison (2010)
24. Leyffer, S.: MacMINLP: Test problems for mixed integer nonlinear programming (2003), http://www.mcs.anl.gov/~leyffer/macminlp
25. Linderoth, J., Kılınç, M.: Personnal communication (2010)
26. Lougee-Heimer, R.: The common optimization interface for operations research. IBM Journal of Research and Development 47, 57–66 (2003), http://www.coin-or.org
27. Marchand, H., Wolsey, L.A.: Aggregation and mixed integer rounding to solve MIPs. Operations Research 49(3), 363–371 (2001)
28. Margot, F.: Testing cut generators for mixed integer linear programming. Mathematical Programming Computation 1, 69–95 (2009)
29. Perregaard, M.: Generative Disjunctive Cuts for Mixed Integer Programs. PhD thesis, Carnegie Mellon University (2003)
30. Sawaya, N., Laird, C.D., Biegler, L.T., Bonami, P., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Lee, J., Lodi, A., Margot, F., Wächter, A.: CMU-IBM open source MINLP project test set (2006), http://egon.cheme.cmu.edu/ibm/page.htm
31. Stubbs, R., Mehrotra, S.: A branch-and-cut method for 0-1 mixed convex programming. Mathematical Programming 86, 515–532 (1999)
32. Wächter, A., Laird, C.D., Kawajir, Y.: Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT (2010), http://www.coin-or.org/Ipopt/documentation/

TSP on Cubic and Subcubic Graphs

Sylvia Boyd[1], René Sitters[2], Suzanne van der Ster[2], and Leen Stougie[2,3,⋆]

[1] School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Canada
sylvia@site.uottawa.ca
[2] Department of Operations Research, VU University Amsterdam, The Netherlands
{rsitters,sster,lstougie}@feweb.vu.nl
[3] CWI, Amsterdam, The Netherlands
stougie@cwi.nl

Abstract. We study the Travelling Salesman Problem (TSP) on the metric completion of cubic and subcubic graphs, which is known to be NP-hard. The problem is of interest because of its relation to the famous 4/3 conjecture for metric TSP, which says that the integrality gap, i.e., the worst case ratio between the optimal values of the TSP and its linear programming relaxation, is 4/3. Using polyhedral techniques in an interesting way, we obtain a polynomial-time 4/3-approximation algorithm for this problem on cubic graphs, improving upon Christofides' 3/2-approximation, and upon the $3/2 - 5/389 \approx 1.487$-approximation ratio by Gamarnik, Lewenstein and Svirdenko for the case the graphs are also 3-edge connected. We also prove that, as an upper bound, the 4/3 conjecture is true for this problem on cubic graphs. For subcubic graphs we obtain a polynomial-time 7/5-approximation algorithm and a 7/5 bound on the integrality gap.

1 Introduction

Given a complete undirected graph $G = (V, E)$ on n vertices with non-negative edge costs $c \in \mathbf{R}^E$, $c \neq 0$, the well-known *Traveling Salesman Problem* (TSP) is to find a Hamiltonian cycle in G of minimum cost. When the costs satisfy the triangle inequality, i.e. when $c_{ij} + c_{jk} \geq c_{ik}$ for all $i, j, k \in V$, we call the problem *metric*. A special case of the metric TSP is the so-called *graph-TSP*, where, given an undirected, unweighted simple underlying graph $G = (V, E)$, a complete graph on V is formed, by defining the cost between two vertices as the number of edges on the shortest path between them, known as the *metric completion* of G.

The TSP is well-known to be NP-hard [20], even for the special cases of graph-TSP. As noticed in [17], APX-hardness follows rather straightforwardly from the APX-hardness of (weighted) graphs with edges of length 1 or 2 ((1,2)-TSP) (Papadimitriou and Yannakakis [22]), even if the maximum degree is 6.

⋆ This research was partially supported by Tinbergen Institute, the Netherlands and the Natural Sciences and Engineering Research Council of Canada.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 65–77, 2011.

In general, the TSP cannot be approximated in polynomial-time to any constant unless $P = NP$, however for the metric TSP there exists the elegant algorithm due to Christofides [9] from 1976 which gives a 3/2-approximation. Surprisingly, in over three decades no one has found an approximation algorithm which improves upon this bound of 3/2, even for the special case of graph-TSP, and the quest for finding such improvements is one of the most challenging research questions in combinatorial optimization. Very recently, Gharan et al.[16] announced a randomized $3/2 - \epsilon$ approximation for graph-TSP for some $\epsilon > 0$.

A related approach for finding approximated TSP solutions is to study the *integrality gap* $\alpha(TSP)$, which is the worst-case ratio between the optimal solution for the TSP problem and the optimal solution to its linear programming relaxation, the so-called *Subtour Elimination Relaxation* (henceforth SER) (see [5] for more details). The value $\alpha(TSP)$ gives one measure of the quality of the lower bound provided by SER for the TSP. Moreover, a polynomial-time constructive proof for value $\alpha(TSP)$ would provide an $\alpha(TSP)$-approximation algorithm for the TSP.

For metric TSP, it is known that $\alpha(TSP)$ is at most 3/2 (see Shmoys and Williamson [24], Wolsey [25]), and is at least 4/3 (a ratio of 4/3 is reached asymptotically by the family of graph-TSP problems consisting of two vertices joined by three paths of length k; see also [5] for a similar family of graphs giving this ratio), but the exact value of $\alpha(TSP)$ is not known. However, there is the following well-known conjecture:

Conjecture 1. For the metric TSP, the integrality gap $\alpha(TSP)$ for SER is 4/3.

As with the quest to improve upon Christofides' algorithm, the quest to prove or disprove this conjecture has been open for almost 30 years, with very little progress made.

A graph is *cubic* if all of its vertices have degree 3, and *subcubic* if they have degree at most 3. A graph is *k-edge connected* if removal of less than k edges keeps the graph connected. A *bridge* in a connected graph is an edge whose removal breaks the graph into two disconnected subgraphs.

In this paper we study the graph-TSP problem on cubic and subcubic graphs. Note that the graphs in the family described above giving a worst-case ratio of 4/3 for $\alpha(TSP)$ are graph-TSPs on bridgeless subcubic graphs. Our main result improves upon Christofides' algorithm by providing a 4/3-approximation algorithm as well as proving 4/3 as an upper bound in Conjecture 1 for the the special case of graph-TSP for which the underlying graph $G = (V, E)$ is a cubic graph. Note that solving the graph-TSP on such graphs would solve the problem of deciding whether a given bridgeless cubic graph G has a Hamilton cycle, which is known to be NP-complete, even if G is also planar (Garey *et al.* [15]) or bipartite (Akiyama *et al.* [2]). In [8] there is an unproven claim that (1,2)-TSP is APX-hard when the graph of edges of length 1 is cubic, which would imply APX-hardness of graph-TSP on cubic graphs.

Also note that the 3/2 ratio of Christofides' algorithm is tight for cubic graph-TSP (see Figure 1). As noted by Gamarnik *et al.* in [14], one approach that can

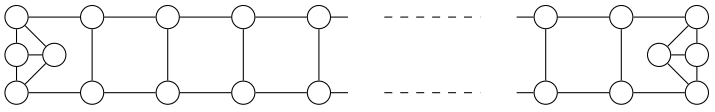


Fig. 1. Example of a cubic graph on which Christofides may attain a ratio of 3/2

be taken for graph-TSP is to look for a polynomial-time algorithm that finds a Hamilton cycle of cost at most τn for some $\tau < 3/2$. Since n is a lower bound for the optimal value for graph-TSP as well as the associated SER[1], this will improve upon Christofides' algorithm by giving a τ-approximation for the graph-TSP, as well as prove that the integrality gap $\alpha(TSP)$ is at most τ for such problems. In [14], Gamarnik *et al.* note a connection between optimal solutions to SER and 3-edge connected cubic graphs. Furthermore, they give an algorithm for graph-TSP where the underlying graph is 3-edge connected and cubic, and for which $\tau = (3/2 - 5/389) \approx 1.487$.

The algorithm of Gamarnik *et al.* provided the first approximation improvement over Christofides' algorithm for the graph-TSP for 3-edge connected cubic graphs. We improve upon their results, both in terms of the value of τ and the class of underlying graphs by proving the following:

Theorem 1. *Every bridgeless simple cubic graph $G = (V, E)$ with $n \geq 6$ has a graph-TSP tour of length at most $\frac{4}{3}n - 2$.*

Our proof of this theorem is constructive, and provides a polynomial-time 4/3-approximation algorithm for graph-TSP on bridgeless cubic graphs. The proof uses polyhedral techniques in a surprising way, which may be more widely applicable. The result also proves that Conjecture 1 is true for this class of TSP problems as an upper bound. The theorem is indeed central in the sense that the other results in this paper are based upon it. One of them is that we show how to incorporate bridges with the same guarantees.

For subcubic graphs it appears to be harder to obtain the same strong results as for cubic graphs. For this class of graph-TSP we obtain a 7/5-approximation algorithm and prove that the integrality gap is bounded by 7/5, still improving considerably over the existing 3/2 bounds. Note that 4/3 is a lower bound for $\alpha(TSP)$ on subcubic graphs.

Relevant literature: Between the first and final submission of this conference paper Aggarwal et al. [1] announced an alternative 4/3 approximation for 3-edge connected cubic graphs. Grigni *et al.* [17] give a polynomial-time approximation scheme (PTAS) for graph-TSP on planar graphs (this was later extended to a PTAS for the weighted planar graph-TSP by Arora *et al.* [3]). For graph G containing a cycle cover with no triangles, Fotakis and Spirakis [12] show that

[1] To see that n is a lower bound for SER, sum all of the so-called "degree constraints" for SER. Dividing the result by 2 shows that the sum of the edge variables in any feasible SER solution equals n.

graph-TSP is approximable in polynomial time within a factor of $17/12 \approx 1.417$ if G has diameter 4 (i.e. the longest path has length 4), and within $7/5 = 1.4$ if G has diameter 3. For graphs that do not contain a triangle-free cycle cover they show that if G has diameter 3, then it is approximable in polynomial time within a factor of $22/15 \approx 1.467$. For graphs with diameter 2 (i.e. TSP(1,2)), a $7/6 \approx 1.167$-approximation for graph-TSP was achieved by Papadimitriou and Yannakakis [22], and improved to $8/7 \approx 1.143$ by Berman and Karpinski [6].

2 Preliminaries

We begin this section with some definitions. Given a graph $G = (V, E)$, we let $V(G)$ denote the vertex set V of G. For any vertex subset $S \subseteq V$, $\delta(S) \subseteq E$, defined as the set of edges connecting S and $V \setminus S$, is called the *cut* induced by S. A cut of cardinality k is called a *k-cut* if it is minimal in the sense that it does not contain any cut as a proper subset.

A *cycle* in a graph is a closed path. In this paper, cycles have no repetition of vertices, which in graph literature is often referred to as *circuits*. A *k-cycle* is a cycle containing k edges, and a *chord* of a cycle of G is an edge not in the cycle, but with both ends u and v in the cycle. A *cycle cover* (also sometimes referred to as 2-factor or perfect 2-matching) of G is a set of vertex disjoint cycles that together span all vertices of G. An *Eulerian subgraph* of G is a connected subgraph where multiple copies of the edges are allowed, and all vertices have even degree. A *perfect matching* M of a graph G is a set of vertex-disjoint edges of G that together span all vertices of G. We call M a *3-cut perfect matching* if every 3-cut of G contains exactly one edge of M.

A well-known theorem of Petersen [23] states that every bridgeless cubic graph contains a perfect matching. Thus the edges of any bridgeless cubic graph can be partitioned into a perfect matching and an associated cycle cover. This idea is important for our main theorem, and we give a useful strengthened form of it below in Lemma 1.

For any edge set $F \subseteq E$, the *incidence vector of* F is the vector $\chi^F \in \mathbf{R}^E$ defined by $\chi^F_e = 1$ if $e \in F$, and 0 otherwise. For any edge set $F \subseteq E$ and $x \in \mathbf{R}^E$, let $x(F)$ denote the sum $\sum_{e \in F} x_e$.

Given graph G, the associated *perfect matching polytope*, $P^M(G)$, is the convex hull of all incidence vectors of the perfect matchings of G, which Edmonds [11] shows to be given by:

$$\begin{aligned} x(\delta(v)) &= 1, \quad && \forall v \in V, \\ x(\delta(S)) &\geq 1, && \forall S \subset V,\ |S| \text{ odd}, \\ 0 \leq x_e &\leq 1, && \forall e \in E. \end{aligned}$$

Using this linear description and similar methods to those found in [19] and [21], we have the following strengthened form of Petersen's Theorem:

Lemma 1. *Let $G = (V, E)$ be a bridgeless cubic graph and let $x^* = \frac{1}{3}\chi^E$. Then x^* can be expressed as a convex combination of incidence vectors of 3-cut perfect matchings, i.e. there exists 3-cut perfect matchings M_i, $i = 1, 2, ..., k$ of G and positive real numbers λ_i, $i = 1, 2, ..., k$ such that*

$$x^* = \sum_{i=1}^{k} \lambda_i(\chi^{M_i}) \text{ and } \sum_{i=1}^{k} \lambda_i = 1. \tag{1}$$

Proof. Since both sides of any 2-cut in a cubic graph have an even number of vertices, it is easily verified that x^* satisfies the linear description above, and thus lies in $P^M(G)$. It follows that x^* can be expressed as a convex combination of perfect matchings of G, i.e. there exist perfect matchings M_i, $i = 1, 2, ..., k$ of G and positive real numbers λ_i, $i = 1, 2, ..., k$ such that (1) holds.

To see that each perfect matching in (1) is a 3-cut perfect matching, consider any 3-cut $\delta(S) = \{e_1, e_2, e_3\}$ of G. Since each side of a 3-cut of any cubic graph must contain an odd number of vertices, any perfect matching must contain 1 or 3 edges of $\delta(S)$. Let $\mathcal{M}_0$ be the set of perfect matchings from (1) that contain all 3 edges of the cut, and let $\mathcal{M}_j$, $j = 1, 2, 3$ be the sets of perfect matchings that contain edge e_j. Define $\alpha_j = \sum_{M_i \in \mathcal{M}_j} \lambda_i$, $j = 0, 1, 2, 3$. Then

$$\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 = x^*(\delta(S)) = 1$$
$$\alpha_0 + \alpha_1 = 1/3,\ \alpha_0 + \alpha_2 = 1/3,\ \alpha_0 + \alpha_3 = 1/3,$$

which implies $\alpha_0 = 0$. □

The perfect matchings $M_i, i = 1, 2, ...k$ of Lemma 1 will be used in the proof of our main theorem in the next section. Note that Barahona [4] provides an algorithm to find for any point in $P^M(G)$ a set of perfect matchings for expressing the point as a convex combination of their incidence vectors in $O(n^6)$ time, and with $k \leq 7n/2 - 1$, for any graph G.

3 Cubic Graphs

In our analysis of the graph-TSP problem for graph $G = (V, E)$, we will consider the equivalent form of the problem, introduced in [10] as the *graphical TSP* of G (henceforth GTSP), in which one seeks a minimum length tour of G in which vertices can be visited more than once and edges can be traversed more than once. The solution, which we refer to as a *GTSP tour*, forms a spanning Eulerian subgraph $H = (V, E')$ of G, which can be transformed into a graph-TSP tour of G of cost $|E'|$ and vice versa. Note that an edge appears at most twice in H.

The idea we will use in the proof of our main theorem is as follows: By Petersen's Theorem we know we can always find a cycle cover of G. Suppose that we can find such a cycle cover that has no more than $n/6$ cycles. Then, contracting the cycles, adding a doubled spanning tree in the resulting graph and uncontracting the cycles would yield a GTSP solution with no more than

$n + 2(n/6 - 1) = 4n/3 - 2$ edges. Together with the obvious lower bound of n on the length of any optimal GTSP tour, this yields an approximation ratio of 4/3. However, such a cycle cover does not always exist (for example, consider the Petersen graph)[2]. Therefore, we take the k cycle covers associated with the 3-cut matchings of Lemma 1 and combine their smaller cycles into larger cycles or Eulerian subgraphs, such as to obtain k covers of G with Eulerian subgraphs which, together with the double spanning tree, result in k GSTP tours with average length at most $4/3n$. For this construction of larger Eulerian subgraphs the following lemma will be useful.

Lemma 2. *Let H_1 and H_2 be connected Eulerian subgraphs of a (sub)cubic graph such that H_1 and H_2 have at least two vertices in common and let H_3 be the* sum *of H_1 and H_2, i.e., the union of their vertices and the sum of their edges, possibly giving rise to parallel edges. Then we can remove two edges from H_3 such that it stays connected and Eulerian.*

Proof. Let u and v be in both subgraphs. The edge set of H_3 can be partitioned into edge-disjoint (u, v)-walks P_1, P_2, P_3, P_4. Vertex u must have two parallel edges which are on different paths, say $e_1 \in P_1$ and $e_2 \in P_2$. When we remove e_1 and e_2 then the graph stays Eulerian. Moreover, it stays connected since u and v are still connected by P_3 and P_4 and, clearly, each vertex on P_1 and P_2 remains connected to either u or v. □

The following lemma, which applies to any graph, allows us to preprocess our graph by removing certain subgraphs.

Lemma 3. *Assume that removing edges $u'u''$ and $v'v''$ from graph $G = (V, E)$ breaks it into two graphs $G' = (V', E')$ and $G'' = (V'', E'')$ with $u', v' \in V'$, and $u'', v'' \in V''$ and such that:*

1. *$u'v' \in E$ and $u'', v'' \notin E$.*
2. *there is a GTSP tour T' in G' of length at most $4|V'|/3 - 2$.*
3. *there is a GTSP tour T'' in $G'' \cup u''v''$ of length at most $4|V''|/3 - 2$.*

Then there is a GTSP tour T in G of length at most $4|V|/3 - 2$.

Proof. If T'' does not use edge $u''v''$ then we take edge $u'u''$ doubled and add tour T'. If T'' uses edge $u''v''$ once then we remove it and add edges $u'u''$, $v'v''$ and $u'v'$ and tour T'. If T'' uses edge $u''v''$ twice then we remove both copies and add edge $u'u''$ doubled, $v'v''$ doubled, and tour T'. □

We use Lemma 3 to remove all subgraphs of the form shown in Figure 2, which we call a *p-rainbow* subgraph. In such subgraphs there is a path $u_0, u_1, \ldots, u_{p+1}$ and path $v_0, v_1, \ldots, v_{p+1}$ for some $p \geq 1$, and a 4-cycle u_0, a, v_0, b with chord ab. Furthermore, there are edges $u_i v_i$ for each $i \in \{1, 2, \ldots, p\}$ but there is no edge

[2] We remark that if G is 3-edge connected and cubic there does exist a triangle- and square-free cycle cover of G which can be found in polynomial time (see [7],[18]), resulting in a straightforward 7/5 approximation algorithm for such graphs.

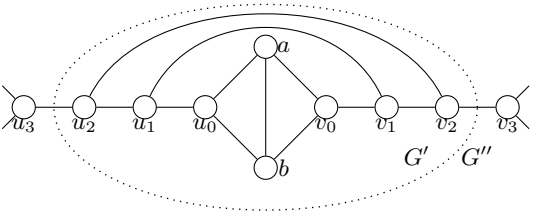


Fig. 2. In this p-rainbow example, $p = 2$ and $u' = u_2, u'' = u_3, v' = v_2$, and $v'' = v_3$

between u_{p+1} and v_{p+1}. The figure shows a p-rainbow for $p = 2$. For general p, the 2-cut of Lemma 3 is given by $u' = u_p$, $u'' = u_{p+1}$, $v' = v_p$, and $v'' = v_{p+1}$. If G contains a p-rainbow G', $p \geq 1$, then we remove G' and add edge $u''v''$ to the remaining graph G''. Note that G'' is also a simple bridgeless cubic graph. We repeat this until there are no more p-rainbows in G'' for any $p \geq 1$. If the final remaining graph G'' has at least 6 vertices, then assuming G'' has a GTSP tour of length at most $4/3|V''| - 2$, we can apply Lemma 3 repeatedly to obtain a GTSP tour of length at most $4/3n - 2$ for the original graph G. If the final remaining graph G'' has less than 6 vertices, then it must have 4 vertices, since it is cubic, hence it forms a complete graph on 4 vertices. In this case we take the Hamilton path from u'' to v'' in G'' and match it with the Hamilton path of the p-rainbow that goes from u_p to v_p to obtain a Hamilton cycle of the graph G'' with the edge $u''v''$ replaced by the p-rainbow. We can then apply Lemma 3 repeatedly to obtain a GTSP tour of length at most $4/3n - 2$ for G.

Proof of Theorem 1. By the above discussion, we assume that there are no p-rainbow subgraphs in G. By Lemma 1 there exist 3-cut perfect matchings $M_1, \ldots, M_k$ and positive real numbers $\lambda_1, \ldots, \lambda_k$ such that $\sum_{i=1}^{k} \lambda_i = 1$ and $\frac{1}{3}\chi^E = \sum_{i=1}^{k} \lambda_i(\chi^{M_i})$. Let $\mathcal{C}_1, \ldots, \mathcal{C}_k$ be the cycle covers of G corresponding to $M_1, M_2, ...M_k$. Since each M_i is a 3-cut perfect matching, each $\mathcal{C}_i$ intersects each 3-cut of G in exactly 2 edges, and hence contains neither a 3-cycle nor a 5-cycle with a chord.

If some $\mathcal{C}_i$ has no more than $n/6$ cycles, then we are done, by the argument given earlier. Otherwise we manipulate each of the cycle covers by operations (i) and (ii) below, which we will show to be well-defined. First operation (i) will be performed as long as possible. Then operation (ii) will be performed as long as possible.

(i) If two cycles C_i and C_j of the cycle cover intersect a (chordless) cycle C of length 4 in G (the original graph) then combine them into a single cycle on $V(C_i) \cup V(C_j)$.

The details of operation (i) are as follows: Assume that u_1u_2 and v_1v_2 are edges of C (and the matching) such that u_1v_1 is an edge of C_i and u_2v_2 is an edge of C_j. Deleting the latter two edges and inserting the former two yields a single cycle of length equal to the sum of the lengths of C_i and C_j. Notice that operation (i)

always leads to cycles of length at least 8. Hence after operation (i) is finished we still have a cycle cover. Operation (ii) below combines cycles into Eulerian subgraphs and subsequently Eulerian subgraphs into larger Eulerian subgraphs, turning the cycle covers into Eulerian subgraph covers. Both types of cover we call simply a *cover* and their elements (cycles and Eulerian subgraphs) we call *components*.

(ii) If two components γ_i and γ_j of the cycle cover or the Eulerian subgraph cover, each having at least 5 vertices, intersect a (chordless) cycle C of length 5 in G (the original graph) then combine them into a single Eulerian subgraph where the number of edges is 1 plus the number of edges of γ_i and γ_j.

The details of operation (ii) are as follows. First note that for any cycle C, its vertex set $V(C)$ has the following (trivial) property:

$\mathcal{P}$: Each $v \in V(C)$ has at least two other vertices $u, w \in V(C)$ such that $vu \in E$ and $vw \in E$.

If two vertex sets both satisfy $\mathcal{P}$ then their union also satisfies $\mathcal{P}$. Since the vertex set of each component γ constructed by operations (i) or (ii) is a result of taking unions of vertex sets of cycles, each such γ has property $\mathcal{P}$. In particular, since G is cubic, this implies that the two components γ_i and γ_j share 2 and 3 vertices with C, respectively (note that they cannot each share exactly 2 vertices, as this would imply that a vertex of C is not included in the cover). We first merge γ_1 and C as in Lemma 2 and remove 2 edges, and then merge the result with γ_2, again removing 2 edges. Altogether we added the 5 edges of C and removed 4 edges.

Operation (ii) leads to Eulerian subgraphs with at least 10 vertices. Thus, any Eulerian subgraph with at most 9 vertices is a cycle. At the completion of operations (i) and (ii), let the resulting Eulerian subgraph covers be $\Gamma_1, \ldots, \Gamma_k$.

Given $\Gamma_1, \ldots, \Gamma_k$, we bound for each vertex its average contribution to the cost of the GTSP tours, weighted by the λ_i's. We define the contribution of a vertex v which in cover Γ_i lies on an Eulerian subgraph with ℓ edges and h vertices as $z_i(v) = \frac{\ell+2}{h}$; the 2 in the numerator is added for the cost of the double edge to connect the component to the others in the GTSP tour. Note that $\sum_{v \in V} z_i(v)$ is equal to the length of the GTSP solution corresponding to Γ_i, plus 2. The average contribution of v over all covers is $z(v) = \sum_i \lambda_i z_i(v)$. When summing this over all vertices v we obtain the average length of the GTSP tours plus 2. We will show that $z(v) \leq 4/3 \ \forall v \in V$.

Observation 1. *For any vertex v and $i \in \{1, 2, \ldots, k\}$, the contribution $z_i(v)$ is*

(a) at most $\frac{h+2}{h}$, where $h = \min\{t, 10\}$ and v is on a cycle of length t in $\mathcal{C}_i$ or after operation (i).
(b) at most 13/10 if operation (ii) was applied to some component containing v.

Proof (Observation 1). Assume that v is on a Eulerian subgraph γ in Γ_i of g vertices. First we prove (*b*). If operation (ii) was applied to some component

containing v, then vertex v was on a cycle of length at least 5 after operation (i). Each application of (ii) adds at least 5 vertices to the component of v. Hence, the number of times that (ii) was applied to the component of v is at most $g/5-1$. Since each application adds exactly one edge, the number of edges in γ is at most $g+g/5-1$. Hence,

$$z_i(v) \leq \frac{g+g/5+1}{g} = \frac{12}{10} + \frac{1}{g} \leq \frac{13}{10}.$$

We use a similar argument to prove (a). Clearly, $g \geq h$. If γ is a cycle then the contribution of v in Γ_i is $(g+2)/g \leq (h+2)/h$ and (a) is true. If γ is not a cycle then this Eulerian subgraph was composed by operation (ii) applied to cycles, each of length at least 5 and one of these had length at least h. Hence, the number of these cycles is at most $1+(g-h)/5$. Since every application of operation (ii) adds one edge extra, the number of edges in γ is at most $g+(g-h)/5$. Hence, since $h \leq 10$,

$$z_i(v) \leq \frac{g+(g-h)/5+2}{g} \leq \frac{g+(g-h)/(h/2)+2}{g} = \frac{h+2}{h}. \qquad \square$$

Note the subtleties in Observation 1: If v is on a cycle of length t in $\mathcal{C}_i$ or after operation (i), and $t \leq 10$, then (a) says that $z_i(v)$ is at most $(t+2)/t$. if $t > 10$, then (a) says that its contribution is at most $12/10$. And finally, if t is 5 or 6 and we know that operation (ii) was applied to some component containing v, then (b) allows us to improve the upper bound on $z_i(v)$ to $13/10$ (for other values of t, (b) does not give an improvement).

From now on we fix any vertex v. Suppose that there is no ℓ such that v is on a 4-cycle or a 5-cycle of Γ_ℓ. Then using Observation 1, we have $z_i(v) \leq \max\{8/6, 13/10\} = 4/3$ for every cover Γ_i, and thus $z(v) \leq 4/3$ and we are done.

Now suppose there exists an ℓ such that v is on a 4-cycle C of Γ_ℓ. Then C must be present in $\mathcal{C}_\ell$ as well. First assume that C is chordless in G. Then all four edges adjacent to C are in the set M_ℓ.

Observation 2. *For any pair of vertices on a chordless cycle of G that appears in any $\mathcal{C}_i$, any path between the two that does not intersect the cycle has length at least 3.*

We partition the set $\mathcal{C}_1, \ldots, \mathcal{C}_k$ according to the way the corresponding M_i's intersect the cycle C. Define sets X_0, X_1, X_2 where $X_j = \{i \mid |C \cap M_i| = j\}$ for $j = 0, 1, 2$. Let $x_t = \sum_{i \in X_t} \lambda_i$, $t = 0, 1, 2$. Clearly $x_0 + x_1 + x_2 = 1$. Since each of the four edges adjacent to C receives total weight $1/3$ in the matchings, we have that $4x_0 + 2x_1 = 4/3 \Rightarrow x_0 = 1/3 - x_1/2$. Since each of the edges of C receives total weight $1/3$ in the matchings, $x_1 + 2x_2 = 4/3 \Rightarrow x_2 = 2/3 - x_1/2$.

Clearly, for any $i \in X_0$, v lies on cycle C in $\mathcal{C}_i$, and thus by Observation 1(a), $z_i(v) \leq 6/4$. By Observation 2, for any $i \in X_1$, v lies on a cycle of length at least 6 in $\mathcal{C}_i$, and thus by Observation 1(a), $z_i(v) \leq 8/6$. For any $i \in X_2$,

if C is intersected by one cycle in $\mathcal{C}_i$, then this cycle has length at least 8 by Observation 2. If for $i \in X_2$, C is intersected by two cycles of length at least 4 each, then, after performing operation (i), v will be on a cycle of length at least 8. Thus using Observation 1(a) one more time, we obtain

$$\begin{aligned} z(v) &\leq x_0 6/4 + x_1 8/6 + x_2 10/8 \\ &= (1/3 - x_1/2)6/4 + x_1 8/6 + (2/3 - x_1/2)10/8 \\ &= 4/3 + x_1(8/6 - 6/8 - 10/16) = 4/3 - x_1/24 \leq 4/3. \end{aligned}$$

We prove now that $z(v) \leq 4/3$ also if C is a 4-cycle with a chord. Let us call the vertices on the cycle u_0, a, v_0, b, let ab be the chord, and v is any of the four vertices. If $u_0v_0 \in E$, then $G = K_4$ (the complete graph on 4 vertices), contradicting the assumption that $n \geq 6$. Thus edges u_0u_1 and v_0v_1 exist, with $u_1, v_1 \notin C$. Notice that $u_1 \neq v_1$ since otherwise G would contain a bridge, contradicting 2-connectedness. Let C' be the cycle containing v in some cycle cover $\mathcal{C}_i$. If C' does not contain edge u_0u_1 then $C' = C$. If, on the other hand, $u_0u_1 \in C'$ then also $v_0v_1 \in C'$ and $ab \in C'$. Note that $u_1v_1 \notin E$ since otherwise we have a p-rainbow subgraph as in Figure 2, and we are assuming that we do not have any such subgraphs. Consequently, C' cannot have length exactly 6. It also cannot have length 7 since then a 3-cut with 3 matching edges would occur. Therefore, any cycle containing u_0u_1 has length at least 8. Applying Observation 1(a) twice we conclude that $z(v) \leq 1/3 \cdot 6/4 + 2/3 \cdot 10/8 = 4/3$.

Now assume there exists a (chordless) 5-cycle C containing v in some Γ_ℓ. Note that we can assume that no $w \in C$ is on a 4-cycle of G, otherwise operation (i) would have been applied and the component of v in Γ_ℓ would have size larger than 5. Note further that C is present in $\mathcal{C}_\ell$ as well. The proof for this case is rather similar to the case for the chordless 4-cycle. Let X_j be the set $\{i \mid |C \cap M_i| = j\}$, for $j = 0, 1, 2$. Let $x_t = \sum_{i \in X_t} \lambda_i$, $t = 0, 1, 2$. Again, we have $x_0 + x_1 + x_2 = 1$. Clearly, for any $i \in X_0$, v lies on C in $\mathcal{C}_i$ and for $i \in X_1$ v lies on a cycle of length at least 7 by Observation 2. Hence, by Observation 1(a) we have $z_i(v) \leq 7/5$ for $i \in X_0$ and $z_i(v) \leq 9/7$ for $i \in X_1$. For any $i \in X_2$ there are two possibilities: Either C is intersected by one cycle in $\mathcal{C}_i$, which, by Observation 2, has length at least 9, or C is intersected in $\mathcal{C}_i$ by two cycles, say C_1 and C_2. In the first case we have $z_i(v) \leq 11/9$ by Observation 1(a). In the second case, as argued before, we can assume that no $w \in C$ is on a 4-cycle of G. Hence, C_1 and C_2 each have at least 5 vertices and operation (ii) will be applied, unless C_1 and C_2 end up in one large cycle by operation (i). In the first case we apply Observation 1(b) and get $z_i(v) \leq 13/10$, and in the second case we apply Observation 1(a): $z_i(v) \leq 12/10$. Hence, for any $i \in X_2$ we have $z_i(v) \leq \max\{11/9, 12/10, 13/10\} = 13/10$.

$$\begin{aligned} z(v) &\leq x_0 7/5 + x_1 9/7 + x_2 13/10 \\ &\leq x_0 7/5 + x_1 13/10 + x_2 13/10 \\ &= x_0 7/5 + (1 - x_0)13/10 = 13/10 + x_0 1/10 \\ &\leq 13/10 + 1/30 = 4/3. \end{aligned}$$

□

As previously mentioned, Barahona [4] provides a polynomial-time algorithm which finds a set of at most $7n/2 - 1$ perfect matchings such that $\frac{1}{3}\chi^E$ can be expressed as a convex combination of the incidence vectors of these matchings. This algorithm runs in $O(n^6)$ time. As shown in the proof of Lemma 1, these matchings will automatically be 3-cut perfect matchings. Once we have this set of perfect matchings then applying operations (i) and (ii) on the corresponding cycle covers gives at least one tour of length at most $4n/3 - 2$ according to the above theorem. As any tour has length at least n for graph-TSP, we have the following approximation result:

Corollary 1. *For graph-TSP on bridgeless cubic graphs there exist a polynomial-time* 4/3 *approximation algorithm.*

As n is a lower bound on the value of SER for graph-TSP it also follows that, as an upper bound, Conjecture 1 is true for this class of problems, i.e.,

Corollary 2. *For graph-TSP on bridgeless cubic graphs the integrality gap for SER is at most* 4/3*.*

We remark that the largest ratio we found so far for $\alpha(TSP)$ on bridgeless cubic examples is 7/6. Without proof we extend the result to include bridges.

Theorem 2. *For a cubic graph with b bridges and s vertices incident to more than one bridge, a TSP tour of length at most $(4/3)(n + b - s) - 2$ can be constructed in polynomial time.*

Since an optimal tour on a graph with b bridges has at least $n + 2b - s$ edges:

Corollary 3. *For graph-TSP on cubic graphs, there exists a polynomial-time* 4/3*-approximation algorithm, and the integrality gap for SER is at most* 4/3*.*

4 Subcubic Graphs

When we allow vertices of degree 2, i.e., we consider 2-connected graphs of maximum degree 3, then the optimal GTSP tour may be as large as $4n/3 - 2/3$. For example, take two vertices joined by three paths of the same length. We conjecture that this bound is tight but have no proof. Instead we can show a bound of $7n/5 - 4/5$, based on relating the cubic graph result to this case. Proofs are omitted from this section.

Theorem 3. *Every 2-edge connected graph of maximum degree 3 has a TSP tour of length at most $\frac{7}{5}n - \frac{4}{5}$.*

As with the cubic case, this result can be extended to include bridges.

Theorem 4. *For a graph of maximum degree 3 consisting of n vertices and b bridges, a TSP tour of length at most $7(n - s + 2b)/5$ can be constructed.*

From the proofs of Theorems 3 and 4 a polynomial-time algorithm can be designed. Since $n + 2b - s$ can be shown to be a lower bound for graph-TSP and for SER on subcubic graphs with b bridges, we have:

Corollary 4. *For graph-TSP on subcubic graphs, there exists a polynomial-time* 7/5*-approximation algorithm, and the integrality gap for SER is at most* 7/5*.*

5 Epilogue

The table below shows the state of knowledge about graph-TSP for various classes of graphs. It contains: (column A) lower bounds on the length of graph-TSP tours on n vertices, for n large enough, (column B) upper bounds on them that we know how to construct, (column C) lower bounds on the integrality gap of SER, (column D) upper bounds on the integrality gap of SER, and (column E) upper bounds on the best possible approximation ratio. We have selected only the bridgeless cases, because they are the crucial ones within the classes. Columns B,D and E in the last two rows comprises our work. The other results, specifically the lower bounds in the last 2 rows are briefly explained in the full version of this paper.

		A-lower	B-upper	C-lb-sep	D-ub-sep	E-apr
1	general, 2-edge connected	$2n-4$	$2n-2$	4/3	3/2	3/2
2	max degree 3, 2 edge conn	$4n/3-2/3$	$7n/5-4/5$	4/3	7/5	7/5
3	cubic 2 edge connected	$11n/9-10/9$	$4n/3-2$	7/6	4/3	4/3

The table shows a gap in our knowledge for each of the problem classes. Closing these gaps is an obvious open research question. Proving APX-hardness of cubic graph-TSP is open as well. Another interesting research question is to improve on the running time of our algorithm, which is highly dominated by the $O(n^6)$-time algorithm of Baharona which works for every graph and every point in the perfect matching polytope. Can we find a faster algorithm for the special case that the graph is cubic and for the special point $\frac{1}{3}\chi^E$? The question is related to the Berge-Fulkerson Conjecture [13] which implies that the point $\frac{1}{3}\chi^E$ can be expressed as the convex combination of at most 6 perfect matchings.

Of course, the main research challenges remain to prove Conjecture 1 or even show a 4/3-approximation algorithm. Also for the special case of graph-TSP this problem is wide open.

References

1. Aggarwal, N., Garg, N., Gupta, S.: A 4/3-approximation for TSP on cubic 3-edge-connected graphs (2011) (manuscript)
2. Akiyama, T., Nishizeki, T., Saito, N.: NP-completeness of the hamiltonian cycle problem for bipartite graphs. Journal of Information Processing 3, 73–76 (1980)
3. Arora, S., Grigni, M., Karger, D., Klein, P., Woloszyn, A.: A polynomial-time approximation scheme for weighted planar graph TSP. In: Proc. of the 9th ACM–SIAM Symposium on Discrete Algorithms, pp. 33–41 (1998)
4. Barahona, F.: Fractional packing of T-joins. SIAM Journal on Discrete Math. 17, 661–669 (2004)

5. Benoit, G., Boyd, S.: Finding the exact integrality gap for small travelling salesman problems. Math. of Operations Research 33, 921–931 (2008)
6. Berman, P., Karpinski, M.: 8/7-approximation algorithm for 1,2-TSP. In: Proc. 17th ACM SIAM Symposium on Discrete Algorithms, pp. 641–648 (2006)
7. Boyd, S., Iwata, S., Takazawa, K.: Finding 2-Factors Covering 3- and 4-Edge Cuts in Bridgeless Cubic Graphs Kyoto University (2010) (manuscript)
8. Csaba, B., Karpinski, M., Krysta, P.: Approximability of dense and sparse instances of minimum 2-connectivity, tsp and path problems. In: Proc. 13th ACM–SIAM Symposium on Discrete Algorithms, pp. 74–83 (2002)
9. Christofides, N.: Worst case analysis of a new heuristic for the traveling salesman problem, Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh (1976)
10. Cornuéjols, G., Fonlupt, J., Naddef, D.: The traveling salesman on a graph and some related integer polyhedra. Math. Programming 33, 1–27 (1985)
11. Edmonds, J.: Maximum matching and a polyhedron with 0,1-vertices. J. of Res. National Bureau of Standards B 69, 125–130 (1965)
12. Fotakis, D., Spirakis, P.: Graph properties that facilitate travelling. Electronic Colloquium on Computational Complexity 31, 1–18 (1998)
13. Fulkerson, D.: Blocking and anti-blocking pairs of polyhedra. Math Programming 1, 168–194 (1971)
14. Gamarnik, D., Lewenstein, M., Sviridenko, M.: An improved upper bound for the TSP in cubic 3-edge-connected graphs. OR Letters 33, 467–474 (2005)
15. Garey, M., Johnson, D., Tarjan, R.: The planar hamiltonian circuit problem is NP-complete. SIAM Journal of Computing 5, 704–714 (1976)
16. Gharan, S.O., Saberi, A., Singh, M.: A Randomized Rounding Approach to the Traveling Salesman Problem (2011) (manuscript)
17. Grigni, M., Koutsoupias, E., Papadimitriou, C.: An approximation scheme for planar graph TSP. In: Proc. 36th Annual Symposium on Foundations of Computer Science, pp. 640–645 (1995)
18. Hartvigsen, D., Li, Y.: Maximum cardinality simple 2-matchings in subcubic graphs, University of Notre Dame (2009) (manuscript)
19. Kaiser, T., Král', D., Norine, S.: Unions of perfect matchings in cubic graphs. Electronic Notes in Discrete Math. 22, 341–345 (2005)
20. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: The Traveling Salesman Problem–A Guided Tour of Combinatorial Optimization. Wiley, Chichester (1985)
21. Naddef, D., Pulleyblank, W.: Matchings in regular graphs. Discrete Math. 34, 283–291 (1981)
22. Papadimitriou, C., Yannakakis, M.: The traveling salesman problem with distances one and two. Math. Oper. Res. 18, 1–11 (1993)
23. Petersen, J.: Die Theorie der regulären graphen. Acta Math. 15, 193–220 (1891)
24. Shmoys, D., Williamson, D.: Analyzing the Held-Karp TSP bound: A monotonicity property with application. Information Processing Letters 35, 281–285 (1990)
25. Wolsey, L.: Heuristic analysis, linear programming and branch and bound. Math. Programming Study 13, 121–134 (1980)

Approximability of Capacitated Network Design

Deeparnab Chakrabarty[1], Chandra Chekuri[2,⋆], Sanjeev Khanna[1,⋆⋆], and Nitish Korula[3,⋆⋆⋆]

[1] Dept. of CIS, University of Pennsylvania, Philadelphia, PA 19104
deepc@seas.upenn.edu, sanjeev@cis.upenn.edu
[2] Dept. of Computer Science, University of Illinois, Urbana, IL 61801
chekuri@cs.illinois.edu
[3] Google Inc., 76 Ninth Ave, 4th Floor, New York, NY 10011
nitish@google.com

Abstract. In the *capacitated* survivable network design problem (Cap-SNDP), we are given an undirected multi-graph where each edge has a capacity and a cost. The goal is to find a minimum cost subset of edges that satisfies a given set of pairwise minimum-cut requirements. Unlike its classical special case of SNDP when all capacities are unit, the approximability of Cap-SNDP is not well understood; even in very restricted settings no known algorithm achieves a $o(m)$ approximation, where m is the number of edges in the graph. In this paper, we obtain several new results and insights into the approximability of Cap-SNDP.

We give an $O(\log n)$ approximation for a special case of Cap-SNDP where the global minimum cut is required to be at least R, by rounding the natural cut-based LP relaxation strengthened with valid knapsack-cover inequalities. We then show that as we move away from global connectivity, the single pair case (that is, when only one pair (s,t) has positive connectivity requirement) captures much of the difficulty of Cap-SNDP: even strengthened with KC inequalities, the LP has an $\Omega(n)$ integrality gap. Furthermore, in directed graphs, we show that single pair Cap-SNDP is $2^{\log^{1-\delta} n}$-hard to approximate for any fixed constant $\delta > 0$.

We also consider a variant of the Cap-SNDP in which multiple copies of an edge can be bought: we give an $O(\log k)$ approximation for this case, where k is the number of vertex pairs with non-zero connectivity requirement. This improves upon the previously known $O(\min\{k, \log R_{\max}\})$-approximation for this problem when the largest minimum-cut requirement, namely $R_{\max}$, is large. On the other hand, we observe that the multiple copy version of Cap-SNDP is $\Omega(\log \log n)$-hard to approximate even for the single-source version of the problem.

⋆ Supported in part by NSF grants CCF-0728782 and CCF-1016684.
⋆⋆ Supported in part by NSF Awards CCF-0635084 and IIS-0904314.
⋆⋆⋆ This work was done while the author was at the Department of Computer Science at the University of Illinios, and was partially supported by NSF grant CCF-0728782 and a University of Illinois Dissertation Completion Fellowship.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 78–91, 2011.

1 Introduction

In this paper we consider the *capacitated* survivable network design problem (Cap-SNDP). The input consists of an undirected n-vertex multi-graph $G(V, E)$ and an integer requirement R_{ij} for each unordered pair of nodes (i, j). Each edge e of G has a cost $c(e)$ and an integer capacity $u(e)$. The goal is to find a minimum-cost subgraph H of G such that for each pair of nodes i, j the capacity of the minimum-cut between i and j in H is at least R_{ij}. This generalizes the well-known survivable network design problem (SNDP) problem in which all edge capacities are 1. SNDP already captures as special cases several fundamental connectivity problems in combinatorial optimization such as the min-cost spanning tree, min-cost Steiner tree and forest, as well as min-cost λ-edge-connected subgraph; each of these problems has been extensively studied on its own and several of these special cases are NP-hard and APX-hard to approximate. Jain, in an influential paper [16], obtained a 2-approximation for SNDP via the standard cut-based LP relaxation using the iterated rounding technique.

Although the above mentioned 2-approximation for SNDP has been known since 1998, the approximability of Cap-SNDP has essentially been wide open even in very restricted special cases. Similar to SNDP, Cap-SNDP is motivated by both practial and theoretical considerations. These problems find applications in the design of resilient networks such as in telecommunication infrastructure. In such networks it is often quite common to have equipment with different discrete capacities; this leads naturally to design problems such as Cap-SNDP. At the outset, we mention that a different and somewhat related problem is also referred to by the same name, especially in the operations research literature. In this version the subgraph H has to support *simultaneously* a flow of R_{ij} between each pair of nodes (i, j); this is more closely related to buy-at-bulk network design [8] and the fixed-charge network flow problems [15]. Our version is more related to connectivity problems such as SNDP.

As far as we are aware, the version of Cap-SNDP that we study was introduced (in the approximation algorithms literature) by Goemans *et al.* [14] in conjunction with their work on SNDP. They made several observations on Cap-SNDP: (i) Cap-SNDP reduces to SNDP if all capacities are the same, (ii) there is an $O(\min(m, R_{\max}))$ approximation where m is the number of edges in G and $R_{\max} = \max_{ij} R_{ij}$ is the maximum requirement, and (iii) if *multiple* copies of an edge are allowed then there is an $O(\log R_{\max})$-approximation. We note that in the capacitated case $R_{\max}$ can be exponentially large in n, the number of nodes of the graph. Carr *et al.* [6] observed that the natural cut-based LP relaxation has an unbounded integrality gap even for the graph consisting of only two nodes s, t connected by parallel edges with different capacities. Motivated by this observation and the goal of obtaining improved approximation ratios for Cap-SNDP, [6] strengthened the basic cut-based LP by using *knapsack-cover* inequalities. (Several subsequent papers in approximation algorithms have fruitfully used these inequalities.) Using these inequalities, [6] obtained a $\beta(G) + 1$ approximation for Cap-SNDP where $\beta(G)$ is the maximum cardinality of a *bond* in the underlying simple graph: a bond is a minimal set of edges that separates

some pair of vertices with positive demand. Although $\beta(G)$ could be $\Theta(n^2)$ in general, this approach gives constant factor approximations for certain topologies of the underlying graph — for instance, a line or a cycle.

The above results naturally lead to several questions. What is the approximability of Cap-SNDP? Should we expect a poly-logarithmic approximation or even a constant factor approximation? If not, what are interesting and useful special cases to consider? And do the knapsack cover inequalities help in the general case? What is the approximability of Cap-SNDP if one allows multiple copies? Does this relaxed version of the problem allow a constant factor approximation?

In this paper we obtain several new positive and negative results for Cap-SNDP that provide new insights into the questions above.

1.1 Our Results

We first discuss results for Cap-SNDP where multiple copies are not allowed. We initiate our study by considering the *global connectivity* version of Cap-SNDP where we want a min-cost subgraph with global min-cut at least R; in other words, there is a "uniform" requirement $R_{ij} = R$ for *all* pairs (i, j). We refer to this as the *Cap-R-Connected Subgraph* problem; the special case when all capacities are unit corresponds to the classical minimum cost λ-edge-connected (spanning) subgraph problem, which is known to be APX-hard [12]. We show the following positive result for arbitrary capacities.

Theorem 1. *There is a randomized $O(\log n)$-approximation algorithm for the Cap-R-Connected Subgraph problem.*

To prove Theorem 1, we begin with a natural LP relaxation for the problem. Almost all positive results previously obtained for the unit capacity case are based on this relaxation. As remarked already, this LP has an unbounded integrality gap even for a graph with two nodes (and hence for Cap-R-Connected Subgraph). We strengthen the relaxation by adding the valid knapsack cover inequalities. Following [6], we find a violated inequality *only if* the current fractional solution does not satisfy certain useful properties. Our main technical tool both for finding a violated inequality and rounding the fractional solution is Karger's theorem on the number of small cuts in undirected graphs [17].

Our approach outlined above may be useful in other network design applications. As a concrete illustration, we use it to solve an interesting and natural generalization of Cap-R-Connected Subgraph, namely, the *k-Way–$\mathcal{R}$-Connected Subgraph* problem. The input consists of $(k-1)$ integer requirements $R_1, \ldots R_{k-1}$, such that $R_1 \leq R_2 \leq \ldots \leq R_{k-1}$. The goal is to find a minimum-cost subgraph H of G such that for each $1 \leq i \leq k-1$, the capacity of any $(i+1)$-way cut of G is at least R_i.That is, the minimum capacity of edges that need to be removed from H to form $(i+1)$ disconnected components must be at least R_i. Note that Cap-R-Connected Subgraph is precisely the k-Way–$\mathcal{R}$-Connected Subgraph, with $k = 2$, and that the k-Way–$\mathcal{R}$-Connected Subgraph problem is not a special case of the general Cap-SNDP as the cut requirements for the

former problem are not expressible as pairwise connectivity constraints. Interestingly, our techniques for Cap-R-Connected Subgraph can be naturally extended to handle the multiway cut requirements, yielding the following generalization of Theorem 1. Furthermore, no better result is known for this problem even in the unit capacity case.

Theorem 2. *There is a randomized $O(k \log n)$-approximation algorithm for the k-Way–$\mathcal{R}$-Connected Subgraph problem with $n^{O(k)}$ running time.*

Once the pairwise connectivity requirements are allowed to vary arbitrarily, the Cap-SNDP problem seems to become distinctly harder. Surprisingly, the difficulty of the general case starts to manifest even for the simplest representative problem in this setting, where there is only one pair (s, t) with $R_{st} > 0$; we refer to this as the *single pair* problem. The only known positive result for this seemingly restricted case is a polynomial-factor approximation that follows from the results in [14,6] for general Cap-SNDP. We give several negative results to suggest that this special case may capture the essential difficulty of Cap-SNDP. First, we show that the single pair problem is $\Omega(\log \log n)$-hard to approximate.

Theorem 3. *The single pair Cap-SNDP problem cannot be approximated to a factor better than $\Omega(\log \log n)$ unless $NP \subseteq DTIME(n^{\log \log \log n})$.*

The above theorem is a corollary of the results in Chuzhoy *et al.* 's work on the hardness of related network design problems [9]. We state it as a theorem to highlight the status of the problem, and defer the proof to the full version [5].

Note that the hardness result above does not rule out a logarithmic/poly-logarithmic approximation, and one might hope to obtain such a result via the LP, strengthened with knapsack cover inequalities. Unfortunately, Carr *et al.* [6] showed that the strengthened LP has integrality gap at least $\lfloor \beta(G)/2 \rfloor + 1$. Thus, new algorithmic techniques are necessary to tackle this problem.

We prove a much stronger negative result for the single pair problem in *directed* graphs. Since in the unit-capacity case, polynomial-time minimum-cost flow algorithms solve the single-pair problem exactly even in directed graphs, the hardness result below shows a stark contrast between the unit-capacity and the non-unit capacity cases.

Theorem 4. *In* directed *graphs, the single pair Cap-SNDP cannot be approximated to a factor better than $2^{\log^{(1-\delta)} n}$ for any $0 < \delta < 1$, unless $NP \subseteq DTIME(n^{\mathtt{polylog}(n)})$. Moreover, this hardness holds for instances in which there are only two distinct edge capacities.*

Allowing Multiple Copies: Given the negative results above for even the special case of the single-pair Cap-SNDP, it is natural to consider the relaxed version of the problem where multiple copies of an edge can be chosen. Specifically, for any integer $\alpha \geq 0$, α copies of e can be bought at a cost of $\alpha \cdot c(e)$ to obtain a capacity $\alpha \cdot u(e)$. In some applications, such as in telecommunication networks, this is a reasonable model. As we discussed, this model was considered by

Goemans *et al.* [14] who gave an $O(\log R_{\max})$ approximation for Cap-SNDP. One can easily obtain an $O(k)$ approximation by taking edges on a single path multiple times for each pair, where k is the number of pairs with $R_{ij} > 0$. When $R_{\max}$ is large, we improve the $\min\{O(k), O(\log R_{\max})\}$-approximation discussed above via the following.

Theorem 5. *In undirected graphs, there is an $O(\log k)$-approximation algorithm for Cap-SNDP with multiple copies, where k is the number of pairs with $R_{ij} > 0$.*

Both our algorithm and analysis are inspired by the $O(\log k)$-competitive online algorithm for the Steiner forest problem by Berman and Coulston [4], and the subsequent adaptation of these ideas for the priority Steiner forest problem by Charikar *et al.* [7]. We complement our algorithmic result by showing that the multiple copy version is $\Omega(\log \log n)$-hard to approximate. This hardness holds even for the *single-source* Cap-SNDP where we are given a source node $s \in V$, and a set of terminals $T \subseteq V$, such that $R_{ij} > 0$ iff $i = s$ and $j \in T$. The following theorem, like Theorem 3, also follows easily from the results of [9].

Theorem 6. *Single source Cap-SNDP with multiple copies cannot be approximated to a factor better than $\Omega(\log \log n)$ unless $NP \subseteq DTIME(n^{\log \log \log n})$.*

Related Work: Network design has a large literature in a variety of areas including computer science and operations research. Practical and theoretical considerations have resulted in numerous models and results. Due to space considerations it is infeasible even to give a good overview of closely related work. We briefly mention some work that allows the reader to compare the model we consider here to related models. As we mentioned earlier, our version of Cap-SNDP is a direct generalization of SNDP and hence is concerned with (capacitated) connectivity between request node pairs. We refer the reader to the survey [18] and some recent and previous papers [14,16,13,10,11,20] for pointers to literature on network design for connectivity.

A different model arises if one wishes to find a min-cost subgraph that supports multicommodity flow for the request pairs; in this model each node pair (i, j) needs to routes a flow of R_{ij} in the chosen graph and these flows simultaneously share the capacity of the graph. We refer to this problem as Capacitated Multicommodity Flow (Cap-MF). Several variants of Cap-MF have been considered: If multiple copies of an edge are allowed, Cap-MF is essentially equivalent to the non-uniform buy-at-bulk network design problem [8]. Buy-at-bulk problems have received substantial attention; we refer the reader to [8] for several pointers to this work. If multiple copies of an edge are not allowed, the approximability of Cap-MF is not well-understood; for example if the flow for each pair is only allowed to be routed on a single path, then even checking feasibility of a given subgraph is NP-Hard since the problem captures the well-known edge-disjoint paths and unsplittable flow problems. Very recently, Andrews, Antonakopoulos and Zhang [1] (among other results) considered the special case of Cap-MF in which the capacities of all edges are *identical*; they obtained a poly-logarithmic

approximation, while allowing poly-logarithmic congestion. (That is, they obtain a bi-criteria approximation, as they may use a poly-logarithmic number of copies of an edge.) When edge capacities are non-uniform, the techniques of [1] do not extend even to the single pair setting, and they leave this as the main open problem for future work. Note that in the single pair case, Cap-SNDP and Cap-MF are identical; as discussed above, we believe that this problem captures much of the difficulty of Cap-SNDP and Cap-MF.

The k-Way–$\mathcal{R}$-Connected Subgraph problem that we consider does not appear to have been considered previously even in the unit-capacity case.

2 The Cap-R-Connected Subgraph Problem

In this section, we prove Theorem 1, giving an $O(\log n)$-approximation for the Cap-R-Connected Subgraph problem. We start by writing a natural linear program relaxation for the problem, and strengthening it using additional valid inequalities, called the *knapsack cover* inequalities. We then show how to round this strengthened LP, obtaining an $O(\log n)$-approximation.

2.1 The Standard LP Relaxation and Knapsack-Cover Inequalities

We assume without any loss of generality that the capacity of any edge is at most R. For each subset $S \subseteq 2^V$, we use $\delta(S)$ to denote the set of edges with exactly one endpoint in S. For a set of edges A, we use $u(A)$ to denote $\sum_{e \in A} u(e)$. We say that a set of edges A satisfies (the cut induced by) S if $u(A \cap \delta(S)) \geq R$. Note that we wish to find the cheapest set of edges which satisfies every subset $\emptyset \neq S \subset V$. The following is the LP relaxation of the standard integer program capturing the problem.

$$\min \sum_{e \in E} c(e)x_e : \quad \forall S \subseteq V, \sum_{e \in \delta(S)} u(e)x_e \geq R, \qquad \forall e \in E,\ 0 \leq x_e \leq 1 \qquad \text{(LP)}$$

(LP) can have integrality gap as bad as R. Consider a graph G on three vertices p, q, r. Edge pq has cost 0 and capacity R; edge qr has cost 0 and capacity $R-1$; and edge pr has cost C and capacity R. To achieve a global min-cut of size at least R, any integral solution must include edge pr, and hence must have cost C. In contrast, in (LP) one can set $x_{pr} = 1/R$, and obtain a total cost of C/R.

In the previous example, any integral solution in which the mincut separating r from $\{p, q\}$ has size at least R must include edge pr, even if qr is selected. The following valid inequalities are introduced precisely to enforce this condition. More generally, let S be a set of vertices, and A be an arbitrary set of edges. Define $R(S, A) = \max\{0, R - u(A \cap \delta(S))\}$ be the *residual* requirement of S that must be satisfied by edges in $\delta(S) \setminus A$. That is, any feasible solution has $\sum_{e \in \delta(S) \setminus A} u(e)x_e \geq R(S, A)$. However, any integral solution also satisfies the following stronger requirement

$$\sum_{e \in \delta(S) \setminus A} \min\{R(S, A), u(e)\}x_e \geq R(S, A)$$

and thus these inequalities can be added to the LP to strengthen it. These additional inequalities are referred to as *Knapsack-Cover* inequalities, or simply KC inequalities, and were first used by [6] in design of approximation algorithms for Cap-SNDP.

Let (LP+KC) denote the standard LP strengthened with all the knapsack cover inequalities.

$$\min \sum_{e \in E} c(e)x_e : \quad \text{(LP) constraints}, \tag{LP+KC}$$

$$\forall A \subseteq E, \forall S \subseteq V, \sum_{e \in \delta(S) \setminus A} \min(u(e), R(S,A))x_e \geq R(S,A) \tag{KC-Inequalities}$$

The Linear Program (LP+KC), like the original (LP), has exponential size. However, unlike the (LP), we do not know of the existence of an efficient separation oracle for this. Nevertheless, as we show below, we do not need to solve (LP+KC); it suffices to get to what we call a *good* fractional solution.

Definition 1. *Given a fractional solution x, we say an edge e is* nearly integral *if $x_e \geq \frac{1}{40 \log n}$, and we say e is* highly fractional *otherwise.*

Definition 2. *For any $\alpha \geq 1$, a cut in a graph G with capacities on edges, is an* α-mincut *if its capacity is within a factor α of the minimum cut of G.*

Theorem 7. [Theorems 4.7.6 and 4.7.7 of [17]] *The number of α-mincuts in an n-vertex graph is at most $n^{2\alpha}$. Moreover, the set of all α-mincuts can be found in $O(n^{2\alpha} \log^2 n)$ time with high probability.*

Given a fractional solution x to the edges, we let A_x denote the set of nearly integral edges, that is, $A_x := \{e \in E : x_e \geq \frac{1}{40 \log n}\}$. Define $\hat{u}(e) = u(e)x_e$ to be the fractional capacity on the edges. Let $\mathcal{S} := \{S \subseteq V : \hat{u}(\delta(S)) \leq 2R\}$. A solution x is called *good* if it satisfies the following three conditions:

(a) The global mincut in G with capacity $\hat{u}$ is at least R, i.e. x satisfies the original constraints.
(b) The KC inequalities are satisfied for the set A_x and the sets in $\mathcal{S}$. Note that if (a) is satisfied, then by Theorem 7, $|\mathcal{S}| \leq n^4$.
(c) $\sum_{e \in E} c(e)x_e$ is at most the value of the optimum solution to (LP+KC).

Note that a good solution need not be *feasible* for (LP+KC) as it is satisfies only a subset of KC-inequalities. We use the ellipsoid method to get such a solution. Such a method was also used in [6], and we defer the details to the full version.

Lemma 1. *There is a randomized algorithm that computes a good fractional solution with high probability.*

2.2 The Rounding and Analysis

Given a good fractional solution x, we now round it to get a $O(\log n)$ approximation to the Cap-R-Connected Subgraph problem. A useful tool for our analysis is the following Chernoff bound (see [19], for instance, for a proof):

Lemma 2. *Let $X_1, X_2, \ldots X_k$ be a collection of independent random variables in $[0,1]$, let $X = \sum_{i=1}^{k} X_i$, and let $\mu = \mathbb{E}[X]$. The probability that $X \leq (1-\delta)\mu$ is at most $e^{-\mu\delta^2/2}$.*

We start by selecting A_x, the set of all nearly integral edges. Henceforth, we lose the subscript and denote the set as simply A. Let $F = E \setminus A$ denote the set of all highly fractional edges; for each edge $e \in F$, select it with probability $(40 \log n \cdot x_e)$. Let $F^* \subseteq F$ denote the set of selected highly fractional edges. The algorithm returns the set of edges $E_A := A \cup F^*$.

It is easy to see that the expected cost of this solution E_A is $O(\log n)$ $\sum_{e \in E} c(e)x_e$, and hence by condition (c) above, within $O(\log n)$ times that of the optimal integral solution. Thus, to prove Theorem 1, it suffices to prove that with high probability, E_A satisfies every cut in the graph G; we devote the rest of the section to this proof. We do this by separately considering cuts of different capacities, where the capacities are w.r.t $\hat{u}$ (recall that $\hat{u}(e) = u(e)x_e$). Let $\mathcal{L}$ be the set of cuts of capacity at least $2R$, that is, $\mathcal{L} := \{S \subseteq V : \hat{u}(\delta(S)) > 2R\}$.

Lemma 3. $\mathbf{Pr}[\ \forall S \in \mathcal{L} :\ u(E_A \cap \delta(S)) \geq R] \ \geq\ 1 - \frac{1}{2n^{10}}$.

Proof. We partition $\mathcal{L}$ into sets $\mathcal{L}_2, \mathcal{L}_3, \cdots$ where $\mathcal{L}_j := \{S \subseteq V : jR < \hat{u}(\delta(S)) \leq (j+1)R\}$. Note that Theorem 7 implies $|\mathcal{L}_j| \leq n^{2(j+1)}$ by condition (a) above. Fix j, and consider an arbitrary cut $S \in \mathcal{L}_j$. If $u(A \cap \delta(S)) \geq R$, then S is clearly satisfied by E_A. Otherwise, since the total $\hat{u}$-capacity of S is at least jR, we have $\hat{u}(F \cap \delta(S)) \geq \hat{u}(\delta(S)) - u(A \cap \delta(S)) \geq (j-1)R$. Thus

$$\sum_{e \in F \cap \delta(S)} \frac{u(e)}{R} x_e \geq (j-1)$$

Recall that an edge $e \in F$ is selected in F^* with probability $(40 \log n \cdot x_e)$. Thus, for the cut S, the expected value of $\sum_{e \in F^* \cap \delta(S)} \frac{u(e)}{R} \geq 40(j-1)\log n$. Since $u(e)/R \leq 1$, we can apply Lemma 2 to get that the probability that S is not satisfied is at most $e^{-16 \log n (j-1)} = 1/n^{16(j-1)}$. Applying the union bound, the probability that there exists a cut in $\mathcal{L}_j$ not satisfied by E_A is at most $n^{2(j+1)}/n^{16(j-1)} = n^{18-14j}$. Thus probability that some cut in $\mathcal{L}$ is not satisfied is bounded by $\sum_{j \geq 2} n^{18-14j} \leq 2n^{-10}$ if $n \geq 2$. Hence with probability at least $1 - 1/2n^{10}$, $A \cup F^*$ satisfies all cuts in $\mathcal{L}$.

One might naturally attempt the same approach for the cuts in $\mathcal{S}$ (recall that $\mathcal{S} = \{S \subseteq V : \hat{u}(\delta(S)) \leq 2R\}$) modified as follows. Consider any cut S, which is partly satisfied by the nearly integral edges A. The fractional edges contribute to the residual requirement of S, and since x_e is scaled up for fractional edges by

a factor of $40 \log n$, one might expect that F^* satisfies the residual requirement, with the $\log n$ factor providing a high-probability guarantee. This intuition is correct, but the KC inequalities are crucial. Consider Example 1; edge pr is unlikely to be selected, even after scaling. In the statement of Lemma 2, it is important that each random variable takes values in $[0, 1]$; thus, to use this lemma, we need the expected capacity from fractional edges to be large compared to the maximum capacity of an individual edge. But the KC inequalities, in which edge capacities are "reduced", enforce precisely this condition. Thus we get the following lemma using a similar analysis as above.

Lemma 4. $\mathbf{Pr}[\ \forall S \in \mathcal{S} :\ u(\delta(E_A \cup \delta(S))) \geq R]\ \geq\ 1 - \frac{1}{n^{12}}$.

The $O(\log n)$-approximation guarantee for the Cap-R-Connected Subgraph problem stated in Theorem 1 follows from the previous two lemmas.

2.3 The k-Way–$\mathcal{R}$-Connected Subgraph Problem.

The k-Way–$\mathcal{R}$-Connected Subgraph problem that we define is a natural generalization of the well-studied min-cost λ-edge-connected subgraph problem. The latter problem is motivated by applications to fault-tolerant network design where any $\lambda - 1$ edge failures should not disconnect the graph. However, there may be situations in which global λ-connectivity may be too expensive or infeasible. For example the underlying graph G may have a single cut-edge but we still wish a subgraph that is as close to 2-edge-connected as possible. We could model the requirement by k-Way–$\mathcal{R}$-Connected Subgraph (in the unit-capacity case) by setting $R_1 = 1$ and $R_2 = 3$; that is, at least 3 edges have to be removed to partition the graph into 3 disconnected pieces.

We briefly sketch the proof of Theorem 2. We work with a generalization of (LP+KC) to i-way cuts, with an original constraint for each $i+1$-way cut, $1 \leq i \leq k-1$, and with KC inequalities added. The algorithm is to select all nearly integral edges e (those with $x_e \geq \frac{1}{40k \log n}$), and select each of the remaining (highly fractional) edges e with probability $40k \log n \cdot x_e$. The analysis is very similar to that of Theorem 1, but we use the following lemma on counting k-way cuts in place of Theorem 7. For details, refer to the full version.

Lemma 5 (Lemma 11.2.1 of [17]). *In an n-vertex undirected graph, the number of k-way cuts with capacity at most α times that of a minimum k-way cut is at most $n^{2\alpha(k-1)}$.*

3 Single-Pair Cap-SNDP in Directed Graphs

In this section, we show that when the underlying graph is directed, single-pair Cap-SNDP is hard to approximate to within a factor of $2^{\log^{(1-\delta)} n}$ for any $\delta > 0$. This proves Theorem 4; our proof proceeds via a reduction from the label cover problem [3].

Definition 3 (*Label Cover Problem).* *The input consists of a bipartite graph $G(A \cup B, E)$ such that the degree of every vertex in A is d_A and degree of every vertex in B is d_B, a set of labels L_A and a set of labels L_B, and a relation $\pi_{(a,b)} \subseteq L_A \times L_B$ for each edge $(a,b) \in E$. Given a labeling $\phi : A \cup B \to L_A \cup L_B$, an edge $e = (a,b) \in E$ is said to be* consistent *iff $(\phi(a), \phi(b)) \in \pi_{(a,b)}$. The goal is to find a labeling that maximizes the fraction of consistent edges.*

The following hardness result for the label-cover problem is a well-known consequence of the PCP theorem [2] and Raz's Parallel Repetition theorem [21].

Theorem 8 ([2,21]). *For any $\epsilon > 0$, there does not exist a poly-time algorithm to decide if a given instance of label cover problem has a labeling where all edges are consistent* (Yes-Instance)*, or if no labeling can make at least $\frac{1}{\gamma}$ fraction of edges to be consistent for $\gamma = 2^{\log^{1-\epsilon} n}$* (No-Instance)*, unless $NP \subseteq DTIME(n^{\mathtt{polylog}(n)})$.*

We now give a reduction from label cover to the single-pair Cap-SNDP in directed graphs. In our reduction, the only non-zero capacity values will be 1, d_A, and d_B. We note that Theorem 8 holds even when we restrict to instances with $d_A = d_B$. Thus our hardness result will hold on single-pair Cap-SNDP instances where there are only two distinct non-zero capacity values.

Given an instance I of the label cover problem with m edges, we create in polynomial-time a directed instance I' of single-pair Cap-SNDP such that if I is a Yes-Instance then I' has a solution of cost at most $2m$, and otherwise, every solution to I' has cost $\Omega(m\gamma^{\frac{1}{4}})$. This establishes Theorem 4 when $\epsilon = \delta/2$.

The underlying graph $G'(V', E')$ for the single-pair Cap-SNDP instance is constructed as follows. The set V' contains a vertex v for every $v \in A \cup B$. We slightly abuse notation and refer to these sets of vertices in V' as A and B as well. Furthermore, for every vertex $a \in A$, and for every label $\ell \in L_A$, the set V' contains a vertex $a(\ell)$. Similarly, for every vertex $b \in B$, and for every label $\ell \in L_B$, the set V' contains a vertex $b(\ell)$. Finally, V' contains a source vertex s and a sink vertex t. The set E' contains the following directed edges:

- For each vertex a in A, we have an arc (s, a) of cost 0 and capacity d_A. For each vertex $b \in B$, there is an arc (b, t) of cost 0 and capacity d_B.
- For each vertex $a \in A$, and for all labels ℓ in L_A, there is an arc $(a, a(\ell))$ of cost d_A and capacity d_A. For each vertex $b \in B$, and for all labels ℓ in L_B, there is an arc $(b(\ell), b)$ of cost d_B and capacity d_B.
- For every edge $(a,b) \in E$, and for every pair of labels $(\ell_a, \ell_b) \in \pi_{(a,b)}$, there is an arc $(a(\ell_a), b(\ell_b))$ of cost 0 and capacity 1.

This completes the description of the network G'. The requirement R_{st} between s and t is m, the number of edges in the label cover instance. It is easy to verify that the size of the graph G' can be constructed in time polynomial in size of G. The lemmas below analyze the cost of Yes-Instance and No-Instance instances; the proofs are deferred to the full version due to space limitation.

Lemma 6. *If the label cover instance is a* Yes-Instance*, then G' contains a subgraph of cost $2m$ which can realize a flow of value m from s to t.*

Lemma 7. *If the label cover instance is a* No-Instance*, then any subgraph of G' that realizes a flow of m units from s to t has cost $\Omega(m\gamma^{\frac{1}{4}})$.*

Since the graph G' can be constructed from G in poly-time, it follows that a poly-time $(\gamma^{1/4}/5)$-approximation algorithm for single-pair Cap-SNDP would give a poly-time algorithm to decide whether a given instance of label cover is a Yes-Instance or a No-Instance contradicting Theorem 8.

4 Cap-SNDP with Multiple Copies Allowed

We now consider the version of Cap-SNDP when multiple copies of any edge e can be chosen. Our algorithm is inspired by the work of Berman and Coulston [4] on online Steiner Forest. For notational convenience, we rename the pairs $(s_1, t_1), \cdots, (s_k, t_k)$, and denote the requirement R_{s_i,t_i} as R_i; the vertices s_i, t_i are referred to as *terminals*. Order the pairs so that $R_1 \geq R_2 \geq \cdots \geq R_k$.

We start with the intuition. The algorithm considers the pairs in decreasing order of requirements, and maintains a *forest solution* connecting the pairs that have been already been processed; that is, if we retain a single copy of each edge in the partial solution constructed so far, we obtain a forest F. For any edge e on the path in F between s_j and t_j, the total capacity of copies of e will be at least R_j. When considering s_i, t_i, we connect them as cheaply as possible, assuming that edges previously selected for F have 0 cost. (Note that this can be done since we are processing the pairs in decreasing order of requirements and for each edge already present in F, the capacity of its copies is at least R_i.) The key step of the algorithm is that *in addition* to connecting s_i and t_i, we also connect the pair to certain other components of F that are "nearby". The cost of these additional connections can be bounded by the cost of the direct connection costs between the pairs. These additional connections are useful in allowing subsequent pairs of terminals to be connected cheaply. In particular, they allow us to prove a $O(\log k)$ upper bound on the approximation factor.

We now describe the algorithm in more detail. The algorithm maintains a forest F of edges that have already been bought; F satisfies the invariant that, after iteration $i-1$, for each $j \leq i-1$, F contains a unique path between s_j and t_j. In iteration i, we consider the pair s_i, t_i. We define the cost function $c_i(e)$ as $c_i(e) := 0$ for edges e already in F, and $c_i(e) := c(e) + \frac{R_i}{u(e)}c(e)$, for edges $e \notin F$. Note that for an edge $e \notin F$, the cost $c_i(e)$ is sufficient to buy enough copies of e to achieve a total capacity of R_i. Thus it suffices to connect s_i and t_i and pay cost $c_i(e)$ for each edge; in the Cap-SNDP solution we would pay at most this cost and get a feasible solution. However, recall that our algorithm also connects s_i and t_i to other "close by" components; to describe this process, we introduce some notation: For any vertices p and q, we use $d_i(p, q)$ to denote the distance between p and q according to the metric given by edge costs $c_i(e)$. We let $\ell_i := d_i(s_i, t_i)$ be the cost required to connect s_i and t_i, given the current solution F. We also define the *class* of a pair (s_j, t_j), and of a component:

- For each $j \leq i$, we say that pair (s_j, t_j) is in *class* h if $2^h \leq \ell_j < 2^{h+1}$. Equivalently, $\texttt{class}(j) = \lfloor \log \ell_j \rfloor$.
- For each connected component X of F, $\texttt{class}(X) = \max_{(s_j,t_j)\in X} \texttt{class}(j)$.

Now, the algorithm connects s_i (respectively t_i) to component X if $d_i(s_i, X)$ (resp. $d_i(t_i, X)$) $\leq 2^{\min\{\texttt{class}(i),\texttt{class}(X)\}}$. That is, if X is close to the pair (s_i, t_i) compared to the classes they are in, we connect X to the pair. As we show in the analysis, this extra connection cost can be charged to some pair (s_j, t_j) in the component X. The complete algorithm description is given below, and this algorithm gives a $O(\log k)$ approximation, proving Theorem 5. The proof can be found in the full version.

CAP-SNDP-MC:
$F \leftarrow \emptyset$ ⟨⟨*F is the forest solution returned*⟩⟩
For $i \leftarrow 1$ to k
  For each edge $e \in F$, $c_i(e) \leftarrow 0$
  For each edge $e \notin F$, $c_i(e) \leftarrow c(e) + (R_i/u(e))c(e)$
  $\ell_i \leftarrow d_i(s_i, t_i)$
  Add to F a shortest path (of length ℓ_i) from s_i to t_i under distances $c_i(e)$
  $\texttt{class}(i) \leftarrow \lfloor \log \ell_i \rfloor$
  For each connected component X of F
    If $d_i(s_i, X) \leq 2^{\min\{\texttt{class}(i),\texttt{class}(X)\}}$
      Add to F a shortest path connecting s_i and X
  For each connected component X of F
    If $d_i(t_i, X) \leq 2^{\min\{\texttt{class}(i),\texttt{class}(X)\}}$
      Add to F a shortest path connecting t_i and X
  Buy $\lceil R_i/u_e \rceil$ copies of each edge e added during this iteration.

5 Conclusions

In this paper we make progress on addressing the approximability of Cap-SNDP. We gave an $O(\log n)$ approximation for the Cap-R-Connected Subgraph problem, which is a capacitated generalization of the well-studied min-cost λ-edge-connected subgraph problem. Can we improve this to obtain an $O(1)$ approximation or prove super-constant factor hardness of approximation? We also highlight the difficulty of Cap-SNDP by focusing on the single pair problem and show hardness results. We believe that understanding the single pair problem is a key step to understanding the general case. In particular, we do not have a non-trivial algorithm even for instances in which the edge capacities are either 1 or U; this appears to capture much of the difficulty of the general problem. As we noted, allowing multiple copies of edges makes the problem easier; in practice, however, it may be desirable to not allow too many copies of an edge to be used. It is therefore of interest to examine the approximability of Cap-SNDP if we allow

only a small number of copies of an edge. Does the problem admit a non-trivial approximation if we allow $O(1)$ copies or, say, $O(\log n)$ copies? This investigation may further serve to delineate the easy versus difficult cases of Cap-SNDP.

Acknowledgements. CC's interest in capacitated network design was inspired by questions from Matthew Andrews. He thanks Matthew Andrews and Lisa Zhang for several useful discussions on their work on capacitated network design for multi-commodity flows.

References

1. Andrews, M., Antonakopoulos, S., Zhang, L.: Minimum-Cost Network Design with (Dis)economies of Scale. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, pp. 585–592. IEEE, Los Alamitos (2010)
2. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM 45(3), 501–555 (1998)
3. Arora, S., Babai, L., Stern, J., Sweedyk, Z.: The Hardness of Approximate Optimia in Lattices, Codes, and Systems of Linear Equations. J. Comp. Sys. Sci. 54(2), 317–331 (1997)
4. Berman, P., Coulston, C.: On-Line Algorithms for Steiner Tree Problems. In: Proceedings, ACM Symposium on Theory of Computation (STOC), pp. 344–353 (1997)
5. Chakrabarty, D., Chekuri, C., Khanna, S., Korula, N.: Approximability of Capacitated Network Design. Technical Report. arXiv:1009.5734
6. Carr, R.D., Fleischer, L.K., Leung, V.J., Phillips, C.A.: Strengthening integrality gaps for capacitated network design and covering problems. In: Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 106–115 (2000)
7. Charikar, M., Naor, J., Schieber, B.: Resource optimization in QoS multicast routing of real-time multimedia. IEEE/ACM Trans. Netw. 12(2), 340–348 (2004)
8. Chekuri, C., Hajiaghayi, M.T., Kortsarz, G., Salavatipour, M.R.: Approximation Algorithms for Non-Uniform Buy-at-Bulk Network Design. In: Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS), pp. 677–686 (2006)
9. Chuzhoy, J., Gupta, A., Naor, J., Sinha, A.: On the approximability of some network design problems. ACM Transactions on Algorithms 4(2) (2008)
10. Chuzhoy, J., Khanna, S.: Algorithms for single-source vertex connectivity. In: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 105–114 (2008)
11. Chuzhoy, J., Khanna, S.: An $O(k^3 \log n)$-Approximation Algorithm for Vertex-Connectivity Survivable Network Design. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, pp. 437–441 (2009)
12. Fernandes, C.G.: A better approximation ratio for the minimum k-edge-connected spanning subgraph problem. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 629–638 (1997)
13. Fleischer, L., Jain, K., Williamson, D.P.: Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. Journal of Computer and System Sciences 72(5), 838–867 (2006)
14. Goemans, M.X., Goldberg, A.V., Plotkin, S.A., Shmoys, D.B., Tardos, É., Williamson, D.P.: Improved Approximation Algorithms for Network Design Problems. In: ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 223–232 (1994)

15. Hewitt, M., Nemhauser, G.L., Savelsbergh, M.W.P.: Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. INFORMS Journal on Computing 22(2), 314–325 (2010)
16. Jain, K.: A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. Combinatorica 21(1), 39–60 (2001)
17. Karger, D.: Random Sampling in Graph Optimization Problems. Ph.D. Thesis, Stanford University (1994)
18. Kortsarz, G., Nutov, Z.: Approximating minimum cost connectivity problems. In: Gonzalez, T.F. (ed.) Handbook of Approximation algorithms and Metaheuristics. CRC Press, Boca Raton (2007)
19. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
20. Nutov, Z.: Approximating minimum cost connectivity problems via uncrossable bifamilies and spider-cover decompositions. In: Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 417–426. IEEE, Los Alamitos (2009)
21. Raz, R.: A parallel repetition theorem. SIAM Journal of Computing 27(3), 763–803 (1998)

Facility Location with Client Latencies: Linear Programming Based Techniques for Minimum Latency Problems*

Deeparnab Chakrabarty[1] and Chaitanya Swamy[2]

[1] Deptartment of CIS, Univ. of Pennsylvania, Philadelphia, PA 19104
deepc@seas.upenn.edu
[2] Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1
cswamy@math.uwaterloo.ca

Abstract. We introduce a problem that is a common generalization of the uncapacitated facility location and minimum latency (ML) problems, where facilities need to be opened to serve clients and also need to be sequentially activated before they can provide service. Formally, we are given a set $\mathcal{F}$ of n facilities with facility-opening costs f_i, a set $\mathcal{D}$ of m clients, connection costs c_{ij} specifying the cost of assigning a client j to a facility i, a root node r denoting the depot, and a time metric d on $\mathcal{F} \cup \{r\}$. Our goal is to open a subset F of facilities, find a path P starting at r and spanning F to activate the open facilities, and connect each client j to a facility $\phi(j) \in F$, so as to minimize $\sum_{i \in F} f_i + \sum_{j \in \mathcal{D}} (c_{\phi(j),j} + t_j)$, where t_j is the time taken to reach $\phi(j)$ along path P. We call this the *minimum latency uncapacitated facility location* (MLUFL) problem.

Our main result is an $O(\log n \cdot \max(\log n, \log m))$-approximation for MLUFL. We also show that any improvement in this approximation guarantee, implies an improvement in the (current-best) approximation factor for group Steiner tree. We obtain *constant* approximations for two natural special cases of the problem: (a) related MLUFL (metric connection costs that are a scalar multiple of the time metric); (b) metric uniform MLUFL (metric connection costs, uniform time-metric). Our LP-based methods are versatile and easily adapted to yield approximation guarantees for MLUFL in various more general settings, such as (i) when the latency-cost of a client is a function of the delay faced by the facility to which it is connected; and (ii) the k-route version, where k vehicles are routed in parallel to activate the open facilities. Our LP-based understanding of MLUFL also offers some LP-based insights into ML, which we believe is a promising direction for obtaining improvements for ML.

1 Introduction

Facility location and vehicle routing problems are two broad classes of combinatorial optimization problems that have been widely studied in the Operations Research community (see, e.g., [15,19]), and have a wide range of applications.

* A full version [4] is available on the CS arXiv.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 92–103, 2011.

Both problems can be described in terms of an underlying set of clients that need to be serviced. In facility location problems, there is a candidate set of facilities that provide service, and the goal is to open some facilities and connect each client to an open facility so as to minimize some combination of the facility-opening and client-connection costs. Vehicle routing problems consider the setting where a vehicle (delivery-man or repairman) provides service, and the goal is to plan a route that visits (and hence services) the clients as quickly as possible. Two common objectives considered are: (i) minimize the total length of the vehicle's route, giving rise to the traveling salesman problem (TSP), and (ii) (adopting a client-oriented approach) minimize the sum of the client delays, giving rise to minimum latency (ML) problems.

These two classes of problems have mostly been considered separately. However, various logistics problems involve both facility-location and vehicle-routing components. For example, consider the following oft-cited prototypical example of a facility location problem: a company wants to determine where to open its retail outlets so as to serve its customers effectively. Now, inventory at the outlets needs to be replenished or ordered (e.g., from a depot); naturally, a customer cannot be served by an outlet unless the outlet has the inventory demanded by it, and delays incurred in procuring inventory might adversely impact customers. Hence, it makes sense for the company to *also* keep in mind the latencies faced by the customers while making its decisions about where to open outlets, which clients to serve at each outlet, and in what order to replenish the open outlets, thereby adding a vehicle-routing component to the problem.

We propose a mathematical model that is a common generalization of the *uncapacitated facility location* (UFL) and *minimum latency* (ML) problems, and abstracts such settings where facilities need to be "*activated*" before they can provide service. Formally, as in UFL, we have a set $\mathcal{F}$ of n facilities, and a set $\mathcal{D}$ of m clients. Opening facility i incurs a *facility-opening cost* f_i, and assigning a client j to a facility i incurs *connection cost* c_{ij}. (The c_{ij}s need not form a metric.) Taking a lead from minimum latency problems, we model activation delays as follows. We have a root (depot) node r, and a *time metric* d on $\mathcal{F} \cup \{r\}$. A feasible solution specifies a subset $F \subseteq \mathcal{F}$ of facilities to open, a path P starting at r and spanning F along which the open facilities are activated, and assigns each client j to an open facility $\phi(j) \in F$. The cost of such a solution is

$$\sum_{i \in F} f_i + \sum_{j \in \mathcal{D}} \left(c_{\phi(j)j} + t_j \right) \tag{1}$$

where $t_j = d_P(r, \phi(j))$ is the time taken to reach facility $\phi(j)$ along path P. One can view c_{ij} as the time facility i takes to serve client j after it has been activated, in which case $(c_{\phi(j),j} + t_j)$ is the delay faced by client j. (Alternatively, if c_{ij} denotes the time taken by a client j to travel to facility i, then the delay faced by j is $\max(c_{\phi(j),j}, t_j)$, which is within a factor 2 of the sum.) We refer to t_j as client j's latency cost. The goal is to find a solution with minimum total cost. We call this the *minimum-latency uncapacitated facility location* (MLUFL) problem.

Apart from being a natural problem of interest, we find MLUFL appealing since it generalizes various diverse problems of interest, in addition to UFL and ML. One such problem, which captures much of the combinatorial core of MLUFL, is what we call the *minimum group latency* (MGL) problem: given an undirected graph with metric edge weights $\{d_e\}$, groups $\{G_j\}$ of vertices, and a root r, the goal is to find a path starting at r that minimizes the sum of the cover times of the groups, where the cover time of G_j is the first time at which some $i \in G_j$ is visited on the path. Observe that MGL can be cast as MLUFL with zero facility costs (where $\mathcal{F} =$ node-set $\setminus \{r\}$), where for each group G_j, we create a client j with $c_{ij} = 0$ if $i \in G_j$ and ∞ otherwise. Note that we may assume that the groups are disjoint (by creating multiple co-located copies of a node), in which case these c_{ij}s form a metric. MGL itself captures various other problems. Clearly, when each G_j is a singleton, we obtain the minimum latency problem. Given a set-cover instance, if we consider a graph whose nodes are (r and) the sets, we create a group G_j for each element j consisting of the sets containing it, and consider the uniform metric, then this MGL problem is simply the *min-sum set cover* (MSSC) problem [9].

Our results and techniques. Our main result is an $O\big(\log n \cdot \max(\log m, \log n)\big)$-approximation algorithm for MLUFL (Section 2.1), which for the special case of MGL, implies an $O(\log^2 n)$ approximation. Complementing this, we show that a ρ-approximation algorithm for MGL yields an $O(\rho \log m)$-approximation algorithm for the *group Steiner tree* (GST) problem [10] on n nodes and m groups. So an improved approximation ratio for MLUFL would yield a corresponding improvement for GST, whose approximation ratio has remained at $O(\log^2 n \log m)$ for a decade [10]. Combined with the result of [14] on the inapproximability of GST, this also shows that MGL, and hence MLUFL with metric connection costs, cannot be approximated to better than a $\Omega(\log m)$-factor unless $NP \subseteq ZTIME\ (n^{\text{polylog}(n)})$.

Given the above hardness result, we investigate certain well-motivated special cases of MLUFL and obtain significantly improved performance guarantees. In Section 2.2, we consider the case where the connection costs form a metric, which is a scalar multiple of the d-metric (i.e., $d_{uv} = c_{uv}/M$, where $M \geq 1$; the problem is trivial if $M < 1$). For example, in a supply-chain logistics problem, this models a natural setting where the connection of clients to facilities, and the activation of facilities both proceed along the same transportation network at different speeds. We obtain a *constant-factor* approximation algorithm for this problem. In Section 2.3, we consider the *uniform MLUFL* problem, which is the special case where the time-metric is uniform. Uniform MLUFL already generalizes MSSC (and also UFL). For uniform MLUFL with metric connection costs (i.e., metric uniform MLUFL), we devise a 10.78-approximation algorithm. (Without metricity, the problem becomes set-cover hard, and we obtain a simple matching $O(\log m)$-approximation.) The chief novelty here lies in the technique used to obtain this result. We give a simple generic reduction (Theorem 4) that shows how to reduce the metric uniform MLUFL problem with facility costs to one *without* facility costs, in conjunction with an algorithm for UFL. This reduction is surprisingly

robust and versatile and yields, for example, $O(1)$-approximations for *metric uniform k-median* (i.e., metric uniform MLUFL where at most k facilities may be opened), and MLUFL with non-uniform latency costs.

We obtain our approximation bounds by rounding the optimal solution to a suitable linear-programming (LP) relaxation of the problem. In Section 3, we leverage this to obtain some interesting insights about the special case of ML, which we believe cast new light on the problem since all previous approximation algorithms for ML are based on combinatorial approaches. In particular, we present an LP-relaxation for ML, and prove that the integrality gap of these relaxations is *upper bounded* by a (small) constant. Our LP is a specialization of our LP-relaxation for MLUFL. Interestingly, the integrality-gap bound for this LP relies only on the fact that the natural LP relaxation for TSP has constant integrality gap. In contrast, the various known algorithms for ML [2,6,1] all utilize algorithms for the arguably harder k-MST problem or its variants. In the full version [4], we describe a second LP relaxation with exponentially-many variables, one for every path (or tree) of a given length bound, where the separation oracle for the dual problem is a rooted path (or tree) orienteering problem: given rewards on the nodes and metric edge costs, find a (simple) path rooted at r of length at most B that gathers maximum reward. We prove that even a bicriteria approximation for the orienteering problem yields an approximation for ML while losing a constant factor. This connection between orienteering and ML is known [7]. But we feel that our alternate proof, where the orienteering problem appears as the separation oracle required to solve the dual LP, offers a more illuminating explanation of the relation between the approximability of the two problems. Our LP-rounding algorithms to prove the constant integrality gaps exploit various ideas developed for scheduling (e.g., α-points) and polyhedral insights for TSP. This suggests that the wealth of LP based machinery could be leveraged for ML as well; we suspect that our LP-relaxations are in fact much better than what we have accounted for.

LP-based techniques tend to be fairly versatile and can be adapted to handle more general variants of the problem. Our algorithms and analyses extend with little effort to handle various generalizations of MLUFL (and hence, ML). One such example (see Section 4) is the setting where the latency-cost of a client is a function (of bounded growth) of the time taken to reach the facility serving it. This yields an approximation algorithm for the $\mathcal{L}_p$-norm generalization of MLUFL, where we take the $\mathcal{L}_p$-norm of the client latencies (instead of the $\mathcal{L}_1$-norm) in the objective function; these norms tradeoff efficiency with fairness making them an appealing measure to consider. Another notable extension is the k-route version, where we may use k paths starting at r to traverse the open facilities.

Related work. There is a vast amount of literature on facility location and vehicle routing; we refer the reader to [4] for a more-detailed discussion of related work. The work that is most closely related to ours is due to Gupta et al. [13], who independently and concurrently also proposed the minimum group latency (MGL) problem (which they arrive at in the course of solving a

different problem), and obtain results similar to ours for MGL. They also obtain an $O(\log^2 n)$-approximation for MGL, and a hardness of approximation for MGL via the reduction from GST to MGL with an $O(\log m)$-factor loss (see also [16]). They reduce MGL to a series of "group orienteering" problems, which they solve using a subroutine due to Charikar et al. [5]. It is not clear how their combinatorial techniques can be extended to handle facility-opening costs in MLUFL.

2 LP-Rounding Approximation Algorithms for MLUFL

We obtain a linear program for MLUFL as follows. We may assume that $d_{ii'}$ is integral for all $i, i' \in \mathcal{F} \cup \{r\}$. Let E denote the edge-set of the complete graph on $\mathcal{F} \cup \{r\}$ and let $d_{max} := \max_{e \in E} d_e$. Let $\mathsf{T} \leq \min\{n, m\} d_{\max}$ be a known upper bound on the maximum activation time of an open facility in an optimal solution. For every facility i, client j, and time $t \leq \mathsf{T}$, we have a variable $y_{i,t}$ indicating if facility i is opened at time t or not, and a variable $x_{ij,t}$ indicating whether client j connects to facility i at time t. Also, for every edge $e \in E$ and time t, we introduce a variable $z_{e,t}$ which denotes if edge e has been traversed *by* time t. Throughout, we use i to index the facilities in $\mathcal{F}$, j to index the clients in $\mathcal{D}$, t to index the time units in $[\mathsf{T}] := \{1, \ldots, \mathsf{T}\}$, and e to index the edges in E.

$$\min \quad \sum_{i,t} f_i y_{i,t} + \sum_{j,i,t} (c_{ij} + t) x_{ij,t} \tag{P}$$

$$\text{s.t.} \quad \sum_{i,t} x_{ij,t} \geq 1 \qquad \forall j; \qquad x_{ij,t} \leq y_{i,t} \quad \forall i, j, t$$

$$\sum_{e} d_e z_{e,t} \leq t \qquad \forall t \tag{2}$$

$$\sum_{e \in \delta(S)} z_{e,t} \geq \sum_{i \in S, t' \leq t} x_{ij,t'} \qquad \forall t, S \subseteq \mathcal{F}, j \tag{3}$$

$$x_{ij,t}, y_{i,t}, z_{e,t} \geq 0 \qquad \forall i, j, t, e; \qquad y_{i,t} = 0 \quad \forall i, t \text{ with } d_{ir} > t.$$

The first two constraints encode that each client is connected to some facility at some time, and that if a client is connected to a facility i at time t, then i must be open at time t. Constraint (2) ensures that at most t "distance" is covered by the tour on facilities by time t, and (3) ensures that if a client is connected to i by time t, then the tour must have visited i by time t. We assume here that $\mathsf{T} = \mathsf{poly}(m)$; this assumption can be removed with a loss of an $(1 + \varepsilon)$ factor (see [4]). Thus, (P) can be solved efficiently since one can efficiently separate over the constraints (3). Let (x, y, z) be an optimal solution to (P), and OPT denote its objective value. For a client j, define $C_j^* = \sum_{i,t} c_{ij} x_{ij,t}$, and $L_j^* = \sum_{i,t} t x_{ij,t}$. We devise various approximation algorithms for MLUFL by rounding (x, y, z).

2.1 An $O(\log n \cdot \max\{\log n, \log m\})$-Approximation Algorithm

We give an overview of the algorithm. Let $N_j = \{i \in \mathcal{F} : c_{ij} \leq 4C_j^*\}$ be the set of facilities "close" to j, and define τ_j as the earliest time t such that

$\sum_{i \in N_j, t' \le t} x_{ij,t'} \ge \frac{2}{3}$. By Markov's inequality, we have $\sum_{i \in N_j} \sum_t x_{ij,t} \ge \frac{3}{4}$ and $\tau_j \le 12 L_j^*$. It is easiest to describe the algorithm assuming first that the time-metric d is a tree metric. Our algorithm runs in phases, with phase ℓ corresponding to time $t_\ell = 2^\ell$. In each phase, we compute a random subtree rooted at r of "low" cost such that for every client j with $\tau_j \le t_\ell$, with constant probability, this tree contains a facility in N_j. To compute this tree, we utilize the rounding procedure of Garg-Konjevod-Ravi (GKR) for the *group Steiner tree* (GST) problem [10] (see Theorem 1 below), by creating a group for each client j with $\tau_j \le t_\ell$ comprising of, roughly speaking, the facilities in N_j. We open all the facilities included in the subtree, and obtain a tour via the standard trick of doubling all edges and performing an Eulerian tour with possible shortcutting. The overall tour is a concatenation of all the tours obtained in the various phases. For each client j, we consider the first tree that contains a facility from N_j (which must therefore be open), and connect j to such a facility.

Given the result for tree metrics, an oft-used idea to handle the case when d is not a tree metric is to approximate it by a distribution of tree metrics with $O(\log n)$ distortion [8]. Our use of this idea is however slightly subtle Instead of moving to a distribution over tree metrics up front, in each phase ℓ, we use the results of [5,8] to *deterministically* obtain a tree $\mathcal{T}_\ell$ with edge weights $\{d_{\mathcal{T}_\ell}(e)\}$, such that the resulting tree metric dominates d and $\sum_{e=(i,i')} d_{\mathcal{T}_\ell}(i,i') z_{e,t_\ell} = O(\log n) \sum_e d_e z_{e,t_\ell}$. This deterministic choice allows to extend our algorithm and analysis effortlessly to the setting where the latency-cost in the objective function is measured by a more general function of the client-latencies. Algorithm 1 is a detailed description of the algorithm. Let $\tau_{\max} = \max_j \tau_j$.

Theorem 1 ([5,8]). *Given any edge weights $\{\mathfrak{z}_e\}_{e \in E}$, one can deterministically construct a weighted tree $\mathcal{T}$ having leaf-set $\mathcal{F} \cup \{r\}$, leading to a tree metric, $d_{\mathcal{T}}(\cdot)$, such that, for any $i, i' \in \mathcal{F} \cup \{r\}$, we have: (i) $d_{\mathcal{T}}(i,i') \ge d_{ii'}$, and (ii) $\sum_{e=(i,i') \in E} d_{\mathcal{T}}(i,i') \mathfrak{z}_{i,i'} = O(\log n) \sum_e d_e \mathfrak{z}_e$.*

Theorem 2 ([10]). *Consider a tree $\mathcal{T}$ rooted at r with n leaves, subsets $G_1, \ldots, G_p$ of leaves, and fractional values $\mathfrak{z}_e$ on the edges of $\mathcal{T}$ satisfying $\mathfrak{z}(\delta(S)) \ge \nu_j$ for every group G_j and node-set S such that $G_j \subseteq S$, where $\nu_j \in \left[\frac{1}{2}, 1\right]$. There exists a randomized polytime algorithm, henceforth called the GKR algorithm, that returns a rooted subtree $T'' \subseteq \mathcal{T}$ such that (i) $\Pr[e \in T''] \le \mathfrak{z}_e$ for every edge $e \in \mathcal{T}$; and (ii) $\Pr[T'' \cap G_j = \emptyset] \le \exp\left(-\frac{\nu_j}{64 \log_2 n}\right)$ for every group G_j.*

Analysis. Consider any phase ℓ. For any subset S of nodes of the corresponding tree $\mathcal{T}'_\ell$ with $r \notin S$, and any $N'_j \subseteq S$ where $j \in D_\ell$, we have $\mathfrak{z}(\delta_{\mathcal{T}'_\ell}(S)) \ge \sum_{i \in N_j, t \le t_\ell} x_{ij,t} \ge 2/3$. This follows from the constraint (3) in the LP. Using Theorem 2, we get the following lemma which bounds the probability of failure in step A1.3.

Lemma 1. *In any phase ℓ, with probability $1 - \frac{1}{\mathsf{poly}(m)}$, we obtain the desired tree T'_ℓ in step A1.3. Also, $\Pr[T'_\ell \cap N'_j \ne \emptyset] \ge 5/9$ for all $j \in D_\ell$.*

Algorithm 1. Given: a fractional solution (x, y, z) to (P)

A1. In each phase $\ell = 0, 1, \ldots, \mathcal{N} := \lceil \log_2(2\tau_{\max}) + 4\log_2 m \rceil$, we do the following. Let $t_\ell = \min\{2^\ell, \mathsf{T}\}$.

A1.1. Use Theorem 1 with edge weights $\{z_{e,t_\ell}\}$ to obtain a tree $\mathcal{T}_\ell = \big(V(\mathcal{T}_\ell), E(\mathcal{T}_\ell)\big)$. Extend $\mathcal{T}_\ell$ to a tree $\mathcal{T}'_\ell$ by adding a dummy leaf edge (i, v_i) of cost f_i to $\mathcal{T}_\ell$ for each facility i. Let $E' = \{(i, v_i) : i \in \mathcal{F}\}$.

A1.2. Map the LP-assignment $\{z_{e,t_\ell}\}_{e \in E}$ to an assignment $\mathfrak{z}$ on the edges of $\mathcal{T}'_\ell$ by setting $\mathfrak{z}_e = \sum_{e \text{ lies on the unique } i\text{-}i' \text{ path in } \mathcal{T}_\ell} z_{ii',t_\ell}$ for all $e \in E(\mathcal{T}_\ell)$, and $\mathfrak{z}_e = \sum_{t \le t_\ell} y_{i,t}$ for all $e = (i, v_i) \in E'$.

A1.3. Define $D_\ell = \{j : \tau_j \le t_\ell\}$. For each client $j \in D_\ell$, we define the group $N'_j = \{v_i : i \in N_j\}$. We now compute a subtree T'_ℓ of $\mathcal{T}'_\ell$ as follows. We obtain $N := \log_2 m$ subtrees $T''_1, \ldots, T''_N$. Each tree T''_r is obtained by executing the GKR algorithm $192\log_2 n$ times on the tree $\mathcal{T}'_\ell$ with groups $\{N'_j\}_{j \in D_\ell}$, and taking the union of all the subtrees returned. Note that we may assume that $i \in T''_r$ iff $(i, v_i) \in T''_r$. Set T'_ℓ to be the first tree in $\{T''_1, \ldots, T''_N\}$ satisfying (i) $\sum_{(i,v_i) \in E(T'_\ell)} f_i \le 40 \cdot 192\log_2 n \sum_{(i,v_i) \in E'} f_i \mathfrak{z}_{i,v_i}$ and (ii) $\sum_{e \in E(T'_\ell) \setminus E'} d_{\mathcal{T}_\ell}(e) \le 40 \cdot 192\log_2 n \sum_{e \in E(\mathcal{T}_\ell)} d_{\mathcal{T}_\ell}(e)\mathfrak{z}_e$; if no such tree exists, the algorithm fails.

A1.4. Now remove all the dummy edges from T'_ℓ, open all the facilities in the resulting tree, and convert the resulting tree into a tour Tour_ℓ traversing all the opened facilities. For every unconnected client j, we connect j to a facility in N_j if some such facility is open (and hence part of Tour_ℓ).

A2. Return the concatenation of the tours Tour_ℓ for $\ell = 0, 1, \ldots, \mathcal{N}$ shortcutting whenever possible. This induces an ordering of the open facilities. If some client is left unconnected, we say that the algorithm has failed.

Since each client j is connected to a facility in N_j, the total connection cost is at most $4\sum_j C^*_j$. Furthermore, from Lemma 1 we get that for every client $j \in D_\ell$, the probability that a facility in N_j is included in the tree T'_ℓ, and hence opened in phase ℓ, is at least $\frac{5}{9}$. The facility-cost incurred in a phase is $O(\log n)\sum_{i,t} f_i y_{i,t}$, and since $\tau_{\max} \le \mathsf{T} = \mathsf{poly}(m)$, the number of phases is $O(\log m)$, so this bounds the facility-opening cost incurred. Also, since the probability that j is not connected (to a facility in N_j) in phase ℓ decreases geometrically (at a rate less than $1/2$) with ℓ when $t_\ell \ge \tau_j$, one can argue that (a) with very high probability (i.e., $1 - 1/\mathsf{poly}(m)$), each client j is connected to some facility in N_j, and (b) the expected latency-cost of j is at most $O(\log n)\sum_{e \in E(\mathcal{T}_\ell)} d_{\mathcal{T}_\ell}(e)\mathfrak{z}_e = O(\log^2 n)\tau_j$.

Lemma 2. *The probability that a client j is not connected by the algorithm is at most $1/m^4$. Let L_j be the random variable equal to j's latency-cost if the algorithm succeeds and 0 otherwise. Then $\mathrm{E}[L_j] = O(\log^2 n)t_{\ell_j}$, where ℓ_j $(= \lceil \log_2 \tau_j \rceil)$ is the smallest ℓ such that $t_\ell \ge \tau_j$.*

Proof. Let P_j be the random phase in which j gets connected; let $P_j := \mathcal{N} + 1$ if j remains unconnected. We have $\Pr[P_j \ge \ell] \le \left(\frac{4}{9}\right)^{(\ell - \ell_j)}$ for $\ell \ge \ell_j$. The algorithm proceeds for at least $4\log_2 m$ phases after phase ℓ_j, so $\Pr[j \text{ is not connected after } \mathcal{N} \text{ phases}] \le 1/m^4$. Now, $L_j \le \sum_{\ell \le P_j} d(\mathsf{Tour}_\ell) \le$

$2\sum_{\ell \le P_j}\sum_{e \in E(T'_\ell)\setminus E'} d_{\mathcal{T}_\ell}(e)$. The RHS is $O(\log n)\sum_{\ell \le P_j}\sum_{e \in E(\mathcal{T}_\ell)} d_{\mathcal{T}_\ell}(e)\mathbb{1}_e = O(\log^2 n)\sum_{\ell \le P_j} t_\ell$ from step A1.3. So $\mathrm{E}\left[L_j\right] = O(\log^2 n)\sum_{\ell=0}^{\mathcal{N}} \Pr[P_j \ge \ell]\cdot t_\ell \le O(\log^2 n)\left[\sum_{\ell=0}^{\ell_j} t_\ell + \sum_{\ell > \ell_j} t_\ell \cdot \left(\frac{4}{9}\right)^{(\ell-\ell_j)}\right] = O(\log^2 n)t_{\ell_j}$.

Theorem 3. *Algorithm 1 succeeds with probability* $1 - 1/\mathsf{poly}(m)$*, and returns a solution of expected cost* $O\big(\log n \cdot \max\{\log n, \log m\}\big)\cdot OPT$.

2.2 MLUFL with Related Metrics

Here, we consider the MLUFL problem when the facilities, clients, and the root r are located in a common metric space that defines the connection-cost metric (on $\mathcal{F} \cup \mathcal{D} \cup \{r\}$), and we have $d_{uv} = c_{uv}/M$ for all $u, v \in \mathcal{F} \cup \mathcal{D} \cup \{r\}$. We call this problem, *related MLUFL*, and design an $O(1)$-approximation algorithm for it.

The algorithm follows a similar outline as Algorithm 1. As before, we build the tour on the open facilities by concatenating tours obtained by "Eulerifying" GST's rooted at r of geometrically increasing length. At a high level, the improvement in the approximation arises because one can now obtain these trees without resorting to Theorem 2 and losing $O(\log n)$-factors in process. Instead, since the d- and c- metrics are related, we obtain a GST on the relevant groups by using a Steiner tree algorithm.

As before, N_j denotes the facilities "close by" (in the c-metric) client j and $\tau_j = O(L^*_j)$. In each phase ℓ we want a GST for the groups N_j for which $\tau_j \le t_\ell$. To obtain this, we first do a facility-location-style clustering of the clients (with $\tau_j \le t_\ell$) to obtain some cluster centers whose N_js are disjoint. We contract these disjoint N_js (of cluster centers) to supernodes and find a minimum Steiner tree connecting these. Since facilities in N_j are close by in the c-metric, and *since the d-metric and c-metric are related*, they are close by in the d-metric as well. Thus, the supernodes in the Steiner tree can be "opened up" to give the GST of not too large cost.

Deciding which facilities to open is tricky since we cannot open facilities in each phase. This is because although N_js are disjoint in an individual phase, they might overlap with N_ks from a different phase. To overcome this, we consider the collection $\mathcal{C}$ of cluster centers created in all the phases, and pick a maximal subset $\mathcal{C}' \subseteq \mathcal{C}$ that yields disjoint N_j's by greedily considering clusters in increasing C^*_j order. We open the cheapest facility i in each of these N_j's, this bounds the facility cost. However, there could be a client $k \in \mathcal{C} \setminus \mathcal{C}'$ which got removed from $\mathcal{C}$ since N_k overlapped with N_j; this k must be connected to i. The issue is that τ_k could be much smaller than τ_j, and thus i needs to be connected to the tree $\mathcal{T}_\ell$ where ℓ is the phase when N_k got connected. To argue this doesn't affect the latency cost too much we once again use the relation between the d-metric and c-metric to show that the total increase in latency cost is at most a constant fraction more.

2.3 MLUFL with a Uniform Time-Metric

We now consider the special case of MLUFL, referred to as *uniform MLUFL*, where the time-metric d is uniform, that is, $d_{ii'} = 1$ for all $i, i' \in \mathcal{F} \cup \{r\}$. When the connection costs form a metric, we call it the *metric uniform MLUFL*. We consider the following simpler LP-relaxation of the problem, where the time t now ranges from 1 to n.

$$\min \quad \sum_{i,t} f_i y_{i,t} + \sum_{j,i,t} (c_{ij} + t) x_{ij,t} \qquad \text{subject to} \qquad \text{(Unif-P)}$$

$$\sum_{i,t} x_{ij,t} \geq 1 \ \forall j; \quad x_{ij,t} \leq y_{i,t} \ \forall i, j, t; \quad \sum_{i} y_{i,t} \leq 1 \ \forall t; \quad x_{ij,t}, y_{i,t} \geq 0 \ \forall i, j, t.$$

The main result of this section is Theorem 4, which shows that a ρ_{UFL}-approximation algorithm for UFL and a γ-approximation algorithm for uniform ZFC MLUFL (uniform MLUFL with zero facility costs) can be combined to yield a $(\rho_{\mathsf{UFL}} + 2\gamma)$-approximation algorithm for metric uniform MLUFL. One can show that $\gamma \leq 9$ (ZFC MLUFL can be reduced to MSSC incurring a constant-factor loss; see [4]), and $\rho_{\mathsf{MLUFL}} \leq 1.5$ [3]; this gives a 19.5 approximation. In the full version, we show that the analysis can be refined to yield an improved 10.773 approximation.

Theorem 4. *Given a ρ_{UFL}-approximation algorithm $\mathcal{A}_1$ for UFL, and a γ-approximation algorithm $\mathcal{A}_2$ for uniform ZFC MLUFL, one can obtain a $(\rho_{\mathsf{UFL}} + 2\gamma)$-approximation algorithm for metric uniform MLUFL.*

Proof. Let $\mathcal{I}$ denote the metric uniform MLUFL instance, and O^* denote the cost of an optimal integer solution. Let $\mathcal{I}_{\mathsf{UFL}}$ be the UFL instance obtained form $\mathcal{I}$ by ignoring the latency costs, and $\mathcal{I}_{\mathsf{ZFC}}$ be the ZFC MLUFL instance obtained from $\mathcal{I}$ by setting all facility costs to zero. Let O^*_{UFL} and O^*_{ZFC} denote respectively the cost of the optimal (integer) solutions to these two instances. Clearly, we have $O^*_{\mathsf{UFL}}, O^*_{\mathsf{ZFC}} \leq O^*$. We use $\mathcal{A}_1$ to obtain a near-optimal solution to $\mathcal{I}_{\mathsf{UFL}}$: let F_1 be the set of facilities opened and let $\sigma_1(j)$ denote the facility in F_1 to which client j is assigned. So we have $\sum_{i \in F_1} f_i + \sum_j c_{\sigma_1(j)j} \leq \rho_{\mathsf{UFL}} \cdot O^*_{\mathsf{UFL}}$. We use $\mathcal{A}_2$ to obtain a near-optimal solution to $\mathcal{I}_{\mathsf{ZFC}}$: let F_2 be the set of open facilities, $\sigma_2(j)$ be the facility to which client j is assigned, and $\pi(i)$ be the position of facility i. So we have $\sum_j \big(c_{\sigma_2(j)j} + \pi(\sigma_2(j))\big) \leq \gamma \cdot O^*_{\mathsf{ZFC}}$.

We now combine these solutions as follows. For each facility $i \in F_2$, let $\mu(i) \in F_1$ denote the facility in F_1 that is nearest to i. We open the set $F = \{\mu(i) : i \in F_2\}$ of facilities. The position of facility $i \in F$ is set to $\min_{i' \in F_2 : \pi(i') = i} \pi(i')$. Each facility in F is assigned a distinct position this way, but some positions may be vacant. Clearly we can always convert the above into a proper ordering of F where each facility $i \in F$ occurs at position $\kappa(i) \leq \min_{i' \in F_2 : \pi(i') = i} \pi(i')$. Finally, we assign each client j to the facility $\phi(j) = \mu(\sigma_2(j)) \in F$. Note that $\kappa(\phi(j)) \leq \pi(\sigma_2(j))$ (by definition). For a client j, we now have $c_{\phi(j)j} \leq c_{\sigma_2(j)\mu(\sigma_2(j))} + c_{\sigma_2(j)j} \leq c_{\sigma_2(j)\sigma_1(j)} + c_{\sigma_2(j)j} \leq c_{\sigma_1(j)j} + 2c_{\sigma_2(j)j}$. Thus, the total cost of the resulting solution is at most $\sum_{i \in F_1} f_i + \sum_j \big(c_{\sigma_1(j)j} + 2c_{\sigma_2(j)j} + \pi(\sigma_2(j))\big) \leq (\rho_{\mathsf{UFL}} + 2\gamma) \cdot O^*$.

3 LP-Relaxations and Algorithms for ML

In this section, we give an LP-relaxation for the ML problem and prove that it has a constant integrality gap. In the full version, we describe another LP-relaxation for ML for which also we prove a constant upper bound on the integrality gap. We believe that our LP-relaxations are stronger than what we have accounted for, and conjecture that the integrality gap of the second LP is at most 3.59, the current best known approximation factor for ML. The second LP also gives an illuminating explanation of the relation between ML and orienteering.

Let $G = (\mathcal{D} \cup \{r\}, E)$ be the complete graph on $N = |\mathcal{D}| + 1$ nodes with edge weights $\{d_e\}$ that form a metric. Let r be the root node at which the path visiting the nodes must originate. We use e to index E and j to index the nodes. We have variables $x_{j,t}$ for $t \geq d_{jr}$ to denote if j is visited at time t, and $z_{e,t}$ to denote (as before) if e has been traversed by time t (where t ranges from 1 to T); for convenience, we think of $x_{j,t}$ as being defined for all t, with $x_{j,t} = 0$ if $d_{j,r} > t$. (As before, one can move to a polynomial-size LP losing a $(1+\epsilon)$-factor.)

$$\min \sum_{j,t} t x_{j,t} \quad \text{subject to} \tag{LP1}$$

$$\sum_t x_{j,t} \geq 1 \ \forall j; \quad \sum_e d_e z_{e,t} \leq t \ \forall t; \quad \sum_{e \in \delta(S)} z_{e,t} \geq \sum_{t' \leq t} x_{j,t'} \ \forall t, S \subseteq \mathcal{D}, j \in S; \quad x, z \geq 0.$$

Theorem 5. *The integrality gap of* (LP1) *is at most* 10.78.

Proof. Let (x, z) be an optimal solution to (LP1), and $L_j^* = \sum_t t x_{j,t}$. For $\alpha \in [0,1]$, define the *α-point of j*, $\tau_j(\alpha)$, to be the smallest t such that $\sum_{t' \leq t} x_{jt'} \geq \alpha$. Let $D_t(\alpha) = \{j : \tau_j(\alpha) \leq t\}$. We round (x, z) as follows. We pick $\alpha \in (0,1]$ according to the density function $q(x) = 2x$. For each time t, using the parsimonious property (see [11]), one can see that $(2z/\alpha)$ is a feasible solution to the sub-tour elimination LP for TSP on the vertices $r \cup D_t(\alpha)$. Then we utilize the $\frac{3}{2}$-integrality-gap of this LP [20,18], to round $\frac{2z}{\alpha}$ and obtain a tour on $\{r\} \cup D_t(\alpha)$ of cost $C_t(\alpha) \leq \frac{3}{\alpha} \cdot \sum_e d_e z_{e,t} \leq \frac{3t}{\alpha}$. We now use Lemma 3 to combine these tours.

Lemma 3 ([12] paraphrased). *Let* $\mathsf{Tour}_1, \ldots, \mathsf{Tour}_k$ *be tours containing* r*, with* Tour_i *having cost* C_i *and containing* N_i *nodes, where* $N_0 := 1 \leq N_1 \leq \ldots \leq N_k = N$*. One can find tours* $\mathsf{Tour}_{i_1}, \ldots, \mathsf{Tour}_{i_b = k}$*, and concatenate them suitably to obtain latency at most* $\frac{3.59}{2} \sum_i C_i (N_i - N_{i-1})$.

The tours we obtain for the different times are nested (as the $D_t(\alpha)$s are nested). So $\sum_{t \geq 1} C_t(\alpha)(|D_t(\alpha)| - |D_{t-1}(\alpha)|) = \sum_{j,t: j \in D_t(\alpha) \setminus D_{t-1}(\alpha)} C_t(\alpha) = \sum_j C_{\tau_j(\alpha)}(\alpha) \leq 3 \sum_j \frac{\tau_j(\alpha)}{\alpha}$. Using Lemma 3, and taking expectation over α (note that $\mathrm{E}\left[\frac{\tau_j(\alpha)}{\alpha}\right] \leq 2L_j^*$), we get total latency cost at most $10.78 \sum_j L_j^*$.

Interestingly, note that in the above proof we did not need any procedure to solve k-MST or its variants, which all previously known algorithms for ML use as a subroutine. Rather, we just needed the integrality gap of the subtour-elimination LP to be a constant.

4 Extensions

Latency cost functions. Consider the setting where the latency-cost of client j is given by λ(time taken to reach the facility serving j), where $\lambda(.)$ is a non-decreasing function; the goal, as before, is to minimize the sum of the facility-opening, client-connection, and client-latency costs. Say that λ has growth at most p if $\lambda(cx) \leq c^p \lambda(x)$ for all $x \geq 0,\ c \geq 1$. It is not hard to see that for concave λ, we obtain the same performance guarantees as those obtained in Section 2. For convex λ, we obtain an $O\big(\max\{(p \log^2 n)^p, p \log n \log m\}\big)$-approximation algorithm for convex latency functions of growth p. As a *corollary*, we obtain an approximation guarantee for $\mathcal{L}_p$-MLUFL, where we seek to minimize the facility-opening cost + client-connection cost + the $\mathcal{L}_p$-norm of client-latencies.

Theorem 6. *There is an $O\big(\max\{(p \log^2 n)^p, p \log n \log m\}\big)$-approximation algorithm for MLUFL with convex monotonic latency functions of growth p. This yields an $O\big(p \log n \max\{\log n, \log m\}\big)$ approximation for $\mathcal{L}_p$-MLUFL.*

In *k-route length-bounded MLUFL*, we are given a budget B and we may use (at most) k paths starting at r of (d-) length at most B to traverse the open facilities and activate them. (So with $B = \infty$, this generalizes the *k-traveling repairmen* problem [7].) Our algorithms easily extend to give: (a) a bicriteria $\big(\text{polylog}, O(\log^2 n)\big)$-approximation for the general k-route MLUFL problem where we violate the budget by a $O(\log^2 n)$ factor; (b) an $(O(1), O(1))$ approximation for MLUFL and ML with related metrics; and (c) a (*unicriterion*) $O(1)$ approximation for metric uniform MLUFL. These guarantees extend to latency functions of bounded growth. In particular, we obtain an $O(1)$ *approximation* for the $\mathcal{L}_p$-norm k-traveling repairmen problem; this is the *first* approximation guarantee for this problem.

References

1. Archer, A., Levin, A., Williamson, D.: A faster, better approximation algorithm for the minimum latency problem. SIAM J. Comput. 37(5), 1472–1498 (2008)
2. Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., Sudan, M.: The Minimum Latency Problem. In: Proc. 26th STOC, pp. 163–171 (1994)
3. Byrka, J.: An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) RANDOM 2007 and APPROX 2007. LNCS, vol. 4627, pp. 29–43. Springer, Heidelberg (2007)
4. Chakrabarty, D., Swamy, C.: Facility Location with Client Latencies: Linear Programming based Techniques for Minimum Latency Problems, `http://arxiv.org/abs/1009.2452`

5. Charikar, M., Chekuri, C., Goel, A., Guha, S.: Rounding via trees: deterministic approximation algorithms for Group Steiner Trees and k-median. In: Proceedings of the 30th STOC, pp. 114–123 (1998)
6. Chaudhuri, K., Godfrey, P.B., Rao, S., Talwar, K.: Paths, Trees and Minimum Latency Tours. In: Proceedings of 44th FOCS, pp. 36–45 (2003)
7. Fakcharoenphol, J., Harrelson, C., Rao, S.: The k-traveling repairman problem. ACM Trans. on Alg. 3(4), Article 40 (2007)
8. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. In: Proc. 35th STOC, pp. 448–455 (2003)
9. Feige, U., Lovász, L., Tetali, P.: Approximating min sum set cover. Algorithmica 40(4), 219–234 (2004)
10. Garg, N., Konjevod, G., Ravi, R.: A polylogarithmic approximation algorithm for the group Steiner tree problem. Journal of Algorithms 37(1), 66–84 (2000)
11. Goemans, M., Bertsimas, D.: Survivable networks, linear programming relaxations and the parsimonious property. Math. Programming 60, 145–166 (1993)
12. Goemans, M., Kleinberg, J.: An improved approximation ratio for the minimum latency problem. In: Proceedings of 7th SODA, pp. 152–158 (1996)
13. Gupta, A., Krishnaswamy, R., Nagarajan, V., Ravi, R.: Approximation Algorithms for Optimal Decision Trees and Adaptive TSP Problems. In: Proceedings of 37th ICALP, pp. 690–701
14. Halperin, E., Krauthgamer, R.: Polylogarithmic inapproximability. In: Proceedings of 35th STOC, pp. 585–594 (2003)
15. Mirchandani, P., Francis, R. (eds.): Discrete Location Theory. John Wiley and Sons, Inc., New York (1990)
16. Nagarajan, V.: Approximation Algorithms for Sequencing Problems. Ph.D. thesis, Tepper School of Business, Carnegie Mellon University (2009)
17. Shmoys, D.B., Tardos, É., Aardal, K.I.: Approximation algorithms for facility location problems. In: Proceedings of 29th STOC, pp. 265–274 (1997)
18. Shmoys, D., Williamson, D.: Analyzing the Held-Karp TSP bound: a monotonicity property with application. Inf. Process. Lett. 35(6), 281–285 (1990)
19. Toth, P., Vigo, D. (eds.): The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia (2002)
20. Wolsey, L.: Heuristic analysis, linear programming and branch and bound. Mathematical Programming Study 13, 121–134 (1980)

An Exact Rational Mixed-Integer Programming Solver

William Cook[1,⋆], Thorsten Koch[2], Daniel E. Steffy[1,⋆], and Kati Wolter[2,⋆⋆]

[1] School of Industrial and Systems Engineering,
Georgia Institute of Technology, Atlanta, GA
bico@isye.gatech.edu, desteffy@gatech.edu
[2] Zuse Institute Berlin, Germany
{koch,wolter}@zib.de

Abstract. We present an exact rational solver for mixed-integer linear programming that avoids the numerical inaccuracies inherent in the floating-point computations used by existing software. This allows the solver to be used for establishing theoretical results and in applications where correct solutions are critical due to legal and financial consequences. Our solver is a hybrid symbolic/numeric implementation of LP-based branch-and-bound, using numerically-safe methods for all binding computations in the search tree. Computing provably accurate solutions by dynamically choosing the fastest of several safe dual bounding methods depending on the structure of the instance, our exact solver is only moderately slower than an inexact floating-point branch-and-bound solver. The software is incorporated into the SCIP optimization framework, using the exact LP solver QSopt_ex and the GMP arithmetic library. Computational results are presented for a suite of test instances taken from the Miplib and Mittelmann collections.

1 Introduction

Mixed-integer programming (MIP) is a powerful and flexible tool for modeling and solving decision problems. Software based on these ideas is utilized in many application areas. Despite their widespread use, few available software packages provide any guarantee of correct answers or certification of results. Possible inaccuracy is caused by the use of floating-point (FP) numbers [14]. FP calculations necessitate the use of built-in tolerances for testing feasibility and optimality, and can lead to calculation errors in the solution of linear-programming (LP) relaxations and in the methods used for creating cutting planes to improve these relaxations.

Due to a number of reasons, for many industrial MIP applications near optimal solutions are sufficient. Cplex, for example, defaults to a relative MIP optimality tolerance of 0.001. Moreover, when data describing a problem arises

⋆ Research supported by NSF Grant CMMI-0726370, ONR Grant N00014-08-1-1104.
⋆⋆ Research funded by DFG Priority Program 1307 “Algorithm Engineering”.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 104–116, 2011.

from imprecise sources, exact feasibility is usually not necessary. Nonetheless, accuracy is important in many settings. Direct examples arise in the use of MIP models to establish fundamental theoretical results and in subroutines for the construction of provably accurate cutting planes. Furthermore, industrial customers of MIP software request modules for exact solutions in critical applications. Such settings include the following.

- Feasibility problems, e.g., chip verification in the VLSI design process [1].
- Compiler optimization, including instruction scheduling [22].
- Combinatorial auctions [21], where serious legal and financial consequences can result from incorrect solutions.

Optimization software relying exclusively on exact rational arithmetic has been observed to be prohibitively slow, motivating the development of more sophisticated techniques to compute exact solutions. Significant progress has been made recently toward computationally solving LP models exactly over the rational numbers using hybrid symbolic/numeric methods [7,10,12,16,17], including the release of the software package QSopt_ex [6]. Exact MIP has seen less computational progress than exact LP, but significant first steps have been taken. An article by Neumaier and Shcherbina [19] describes methods for safe MIP computation, including strategies for generating safe LP bounds, infeasibility certificates, and cutting planes. The methods they describe involve directed rounding and interval arithmetic with FP numbers to avoid incorrect results.

The focus of this article is to introduce a hybrid branch-and-bound approach for exactly solving MIPs over the rational numbers. Section 2 describes how rational and safe FP computation can be coupled together, providing a fast and general framework for exact computation. Section 3 describes several methods for computing valid LP bounds, which is a critical component of the hybrid approach. Section 4 describes an exact branch-and-bound implementation within SCIP [1,2] and includes detailed computational results on a range of test libraries comparing different dual bounding strategies. The exact solver is compared with an inexact branch-and-bound solver and observed to be only moderately slower.

2 Hybrid Rational/Safe Floating-Point Approach

Two ideas for exact MIP proposed in the literature, and tested to some extent, are the *pure rational approach* [7] and the *safe-FP approach* [9,19]. Both utilize LP-based branch-and-bound. The difference lies in how they ensure the computed results are correct.

In the pure rational approach, correctness is achieved by storing the input data as rational numbers, by performing all arithmetic operations over the rationals, and by applying an exact LP solver [12] in the dual bounding step. This approach is especially interesting because it can handle a broad class of problems: MIP instances described by rational data. However, replacing all FP operations by rational computation will increase running times noticeably. For example, while the exact LP solver QSopt_ex avoids many unnecessary rational computations

and is efficient on average, Applegate et al. [7] observed a greater slowdown when testing an exact MIP solver that relied on rational arithmetic and called QSopt_ex for each node LP computation.

In order to limit the degradation in running time, the idea of the safe-FP approach is to continue to use FP-numbers as much as possible, particularly within the LP solver. However, extra work is necessary to ensure correct decisions in the branch-and-bound algorithm. Correctness of certain computations can be ensured by controlling the rounding mode for FP operations. Valid dual bounds can often be obtained by post-processing approximate LP solutions; this type of safe dual bounding technique has been successfully implemented in Concorde [5] for the traveling salesman problem. A generalization of the method for MIPs is described in [19]. Furthermore, the idea of manipulating the rounding mode can be applied to cutting-plane separation. In [9], this idea was used to generate numerically safe Gomory mixed-integer cuts. Nevertheless, whether the safe-FP approach leads to acceptable running times for general MIPs has not been investigated. Although the safe-FP version of branch-and-bound has great advantages in speed over the pure rational approach, it has several disadvantages. Everything, including input data and primal solutions, is stored as FP numbers. Therefore, correct results can only be ensured for MIP instances that are given by FP-representable data and that have a FP-representable optimal solution if they are feasible. Some rationally defined problems can be scaled to have FP-representable data. However, this is not always possible due to the limited representation of floating-point numbers, and the resulting large coefficients can lead to numerical difficulties. The applicability is even further limited as the safe dual bounding method discussed in [19] requires, in general, lower and upper bounds on all variables. Weakness in the safely generated bound values may also increase the number of nodes processed by the branch-and-bound solver. Additionally, due to numerical difficulties, some branch-and-bound nodes may only be processable by an exact LP solver.

To summarize, the pure rational approach is always applicable but introduces a large overhead in running time while the safe-FP approach is more efficient but of limited applicability.

Since we want to solve MIPs that are given by rational data efficiently and exactly we have developed a version of branch-and-bound that attempts to combine the advantages of the pure rational and safe-FP approaches, and to compensate for their individual weaknesses. The idea is to work with two branch-and-bound processes. The *main process* implements the rational approach. Its result is surely correct and will be issued to the user. The other one serves as a *slave process*, where the faster safe-FP approach is applied. To achieve reasonable running time, whenever possible the expensive rational computation of the main process will be skipped and certain decisions from the faster safe-FP process will be substituted. In particular, safe dual bound computations in the slave process can often replace exact LP solves in the main process. The rational process provides the exact problem data, allows to correctly store primal solutions, and makes exact LP solves possible whenever needed.

Algorithm 1. Branch-and-bound for exactly solving MIPs

Input: (MIP) $\max\{c^T x : x \in P\}$ with $P := \{x \in \mathbb{R}^n : Ax \leq b, x_i \in \mathbb{Z} \text{ for all } i \in I\}$, $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, and $I \subseteq \{1, \ldots, n\}$.

Output: *Exact* optimal solution $x^\star$ of MIP with objective value $c^\star$ or conclusion that MIP is infeasible ($c^\star = -\infty$).

1. *FP-problem:* Store (FP-MIP) $\max\{\widetilde{c}^T x : x \in \widetilde{P}\}$ with $\widetilde{P} := \{x \in \mathbb{R}^n : \widetilde{A}x \leq \widetilde{b}, x_i \in \mathbb{Z} \text{ for all } i \in I\}$, $\widetilde{A} \in \mathbb{M}^{m \times n}$, $\widetilde{b} \in \mathbb{M}^m$, and $\widetilde{c} \in \mathbb{M}^n$.
2. *Init:* Set $\mathcal{L} := \{(P, \widetilde{P})\}$, $L := -\infty$, x^{MIP} to be empty, and $c^{\text{MIP}} := -\infty$.
3. *Abort:* If $\mathcal{L} = \emptyset$, stop and return x^{MIP} and c^{MIP}.
4. *Node selection:* Choose $(P_j, \widetilde{P}_j) \in \mathcal{L}$ and set $\mathcal{L} := \mathcal{L} \setminus \{(P_j, \widetilde{P}_j)\}$.
5. *Dual bound:* Solve LP-relaxation $\max\{\widetilde{c}^T x : x \in \widetilde{LP}_j\}$ *approximately.*
   (a) If $\widetilde{LP}_j$ is *claimed* to be empty, *safely* check if LP_j is empty.
      i. If LP_j is empty, set $c^\star := -\infty$.
      ii. If LP_j is not empty, solve LP-relaxation $\max\{c^T x : x \in LP_j\}$ *exactly.* Let $x^\star$ be an *exact* optimal LP solution and $c^\star$ its objective value.
   (b) If $\widetilde{LP}_j$ is *claimed* not to be empty, let $x^\star$ be an *approximate* optimal LP solution and compute a *safe* dual bound $c^\star$ with $\max\{c^T x : x \in LP_j\} \leq c^\star$.
6. *Bounding:* If $\overline{c^\star} \leq L$, goto Step 3.
7. *Primal bound:*
   (a) If $x^\star$ is *approximate* LP solution and claimed to be feasible for FP-MIP, solve LP-relaxation $\max\{c^T x : x \in LP_j\}$ *exactly.* If LP_j is *in fact* empty, goto Step 3. Otherwise, let $x^\star$ be an *exact* optimal LP solution and $c^\star$ its objective value.
   (b) If $x^\star$ is *exact* LP solution and *truly* feasible for MIP:
      i. If $c^\star > c^{\text{MIP}}$, set $x^{\text{MIP}} := x^\star$, $c^{\text{MIP}} := c^\star$, and $L := \underline{c^\star}$.
      ii. Goto Step 3.
8. *Branching:* Choose index $i \in I$ with $x_i^\star \notin \mathbb{Z}$.
   (a) Split P_j in $Q_1 := P_j \cap \{x : x_i \leq \lfloor x_i^\star \rfloor\}$, $Q_2 := P_j \cap \{x : x_i \geq \lceil x_i^\star \rceil\}$.
   (b) Split $\widetilde{P}_j$ in $\widetilde{Q}_1 := \widetilde{P}_j \cap \{x : x_i \leq \lfloor x_i^\star \rfloor\}$, $\widetilde{Q}_2 := \widetilde{P}_j \cap \{x : x_i \geq \lceil x_i^\star \rceil\}$.
   Set $\mathcal{L} := \mathcal{L} \cup \{(Q_1, \widetilde{Q}_1), (Q_2, \widetilde{Q}_2)\}$ and goto Step 3.

The complete procedure is given in Alg. 1. The set of FP-representable numbers is denoted by $\mathbb{M}$; lower and upper approximations of $x \in \mathbb{Q}$ are denoted $\underline{x} \in \mathbb{M}$ and $\overline{x} \in \mathbb{M}$, respectively. The slave process, which utilizes the safe-FP approach, works on a MIP instance with FP-representable data. It is set up in Step 1 of the algorithm. If the input data are already FP-representable, both processes solve the same MIP instance, i.e., $\widetilde{P} := P$ and $\widetilde{c} := c$ in Step 1. In principle, this results in employing only the safe-FP approach except for some necessary exact LP solves. Otherwise, an approximation of the MIP with $P \approx \widetilde{P}$, $c \approx \widetilde{c}$ or a relaxation with $P \subseteq \widetilde{P}$, $c = \widetilde{c}$ is constructed; called *FP-approximation* and *FP-relaxation*, respectively. The choice depends on the dual bounding method applied in the slave process (see Sect. 3).

On the implementation side, we maintain only a single branch-and-bound tree. At the root node of this common tree, we store the LP relaxations of both processes: $\max\{c^T x : x \in LP\}$ and $\max\{\widetilde{c}^T x : x \in \widetilde{LP}\}$. In addition, for each node, we know the branching constraint that was added to create the subproblem in both processes. Branching on variables, performed in Step 8, introduces the same bounds for both processes.

The use of primal and dual bounds to discard subproblems (see Steps 5, 6, and 7) is a central component of the branch-and-bound algorithm. In particular, in the exact MIP setting, the efficiency strongly depends on the strength of the dual bounds and the time spent generating them (Step 5). The starting point of this step is the *approximate LP solution* of the slave process. It is obtained by an LP solver that works on FP-numbers and accepts rounding errors; referred to as *inexact LP solver*. Depending on the result, we safely check whether the rational LP, i.e., the exact LP relaxation, is also infeasible or we compute a safe dual bound by post-processing the approximate LP solution. Different techniques are discussed in Sect. 3 and are computationally evaluated in Sect. 4. We only perform the exact but expensive dual bound computation of the main process if it is necessary (Step 5(a)ii).

Dual and primal bounds are stored as FP-numbers and the bounding in Step 6 is performed without tolerances; a computed bound that is not FP-representable is relaxed in order to be safe. For the primal (lower) bound L, this means $L < c^{\text{MIP}}$ if the objective value c^{MIP} of the incumbent solution x^{MIP} is not in $\mathbb{M}$.

Algorithm 1 identifies primal solutions by checking LP solutions for integrality. This check, performed in Step 7, depends on whether the LP was already solved exactly at the current node. If so, we exactly test the integrality of the exact LP solution. Otherwise, we decide if it is worth solving the LP exactly. We deem it worthy if the approximate LP solution is nearly integral. In this case, we solve the LP exactly, using the corresponding basis to warm start the exact LP solver (hopefully with few pivots and no need to increase the precision) and perform the exact integrality test on the exact LP solution. In order to correctly report the optimal solution found at the end of Step 3, the incumbent solution (that is, the best feasible MIP solution found thus far) and its objective value are stored as rational numbers.

This hybrid approach can be extended to a branch-and-cut algorithm with primal heuristics and presolving; but the focus of this article is on the development of the basic branch-and-bound framework.

3 Safe Dual Bound Generation Techniques

This section describes several methods for computing valid LP dual bounds. The overall speed of the MIP solver will be influenced by several aspects of the dual bounding strategy; how generally applicable the method is, how quickly the bounds can be computed and how strong the bounds are, because weak bounds can increase the node count.

3.1 Exact LP Solutions

The most straightforward way to compute valid LP bounds is to solve each node LP relaxation exactly. This strategy is always applicable and produces the tightest possible bounds. Within a branch-and-bound framework the dual simplex algorithm can be warm started with the final basis computed at the parent node, speeding up the LP solution process. The fastest exact rational LP solver currently available is QSOPT_EX [7]. The strategy used by this solver can be summarized as follows: the basis returned by a double-precision LP solver is tested for optimality by symbolically computing the corresponding basic solution, if it is suboptimal then additional simplex pivots are performed with an increased level of precision and this process is repeated until the optimal basis is identified. This method is considerably faster than using rational arithmetic exclusively and is usually no more than two to five times slower than inexact LP solvers. For problems where the basis determined by the double-precision subroutines of QSOPT_EX is not optimal, the additional increased precision simplex pivots and additional exact basic solution computations significantly increase the solution time.

3.2 Basis Verification

Any exactly feasible dual solution provides a valid dual bound. Therefore, valid dual bounds can be determined by symbolically computing the dual solution corresponding to a numerically obtained LP basis, without performing the extra steps required to identify the exact optimal solution. If the dual solution is feasible, its objective value gives a valid bound. If it is infeasible, an infinite bound is returned. Within the branch-and-bound algorithm, an infinite dual bound will lead to more branching. Due to the fixing of variables, branching often remediates numerical problems in the LP relaxations. This strategy avoids the extended precision simplex pivoting that can occur when solving each node LP exactly, but it can increase the number of nodes.

3.3 Primal-Bound-Shift

Valid bounds can also be produced by correcting approximate dual solutions. A special case occurs when all primal variables have finite upper and lower bounds. The following technique was employed by Applegate et. al. in the CONCORDE software package [5] and is described more generally for MIPs by Neumaier and Shcherbina [19]. Consider a primal problem of the form $\max\{c^T x : Ax \leq b, 0 \leq x \leq u\}$ with dual $\min\{b^T y + u^T z : A^T y + z \geq c,\ y, z \geq 0\}$. Given an approximate dual solution $\tilde{y}, \tilde{z}$, an exactly feasible dual solution $\hat{y}, \hat{z}$ is constructed by setting $\hat{y}_i = \max\{0, \tilde{y}_i\}$ and $\hat{z}_i = \max\{0, (c - A^T \hat{y})_i\}$. This gives the valid dual bound $b^T \hat{y} + u^T \hat{z}$. When working with a FP-relaxation of the original problem, this bound can be computed using floating-point arithmetic with safe directed rounding to avoid the symbolic computation of the dual feasible solution. The simplicity of computing this bound suggests it will be an excellent choice when applicable. However, if some primal variable bounds are large or missing it may produce weak or infinite bounds, depending on the feasibility of $\tilde{y}, \tilde{z}$.

3.4 Project-and-Shift

Correcting an approximate dual solution to be exactly feasible in the absence of primal variable bounds is still possible. Consider a primal problem of the form $\max\{c^T x : Ax \leq b\}$ with dual $\min\{b^T y : A^T y = c,\ y \geq 0\}$. An approximate dual solution $\tilde{y}$ can be corrected to be feasible by projecting it into the affine hull of the dual feasible region and then shifting it to satisfy all of the non-negativity constraints, while maintaining feasibility of the equality constraints. These operations could involve significant computation if performed on a single LP. However, under some assumptions, the most time consuming computations can be performed only once at the root node of the branch-and-bound tree and reused for each node bound computation. The root node computations involve solving an auxiliary LP exactly and symbolically LU factoring a matrix; the cost of each node bound computation is dominated by performing a back-solve of a pre-computed symbolic LU factorization, which is often faster than solving a node LP exactly. This method is more generally applicable than the primal-bound-shift method, but relies on some conditions that are met by most, but not all, of the problems in our test set. A detailed description and computational study of this algorithm can be found in [20]. A related method is also described by Althaus and Dumitriu [4].

3.5 Combinations and Beyond

The ideal dual bounding method is generally applicable, produces tight bounds, and computes them quickly. Each of the four methods described so far represents some trade-off between these conflicting characteristics. The exact LP method is always applicable and produces the tightest possible bounds, but is computationally expensive. The primal-bound-shift method computes valid bounds very quickly, but relies on problem structure that may not always be present. The basis verification and project-and-shift methods provide compromises in between, with respect to speed and generality. Also, since the relative performance of these dual bounding methods strongly depends on the (sub)problem structure it may change throughout the tree. Therefore, a strategy that combines and switches between the bounding techniques is the best choice for an exact MIP solver intended to efficiently solve a broad class of problems.

In Sect. 4, we will evaluate the performance of each dual bounding method presented here and analyze in what situations which technique works best. In a final step, we then study different strategies to automatically decide how to compute safe dual bounds for a given MIP instance. The central idea is to apply fast primal-bound-shift as often as possible and if necessary employ another method depending on the problem structure. In this connection, we will address the question of whether this decision should be static or dynamic.

In the first version ("Auto"), we decide on the method dynamically in Step 5. At each node primal-bound-shift is applied, and in case it produces an infinite bound one of the other methods is applied. The drawbacks are that it allows for unnecessary computations and that it requires an FP-relaxation for

the slave process in order to support primal-bound-shift. Alternatively, we can guess whether primal-bound-shift will work ("Auto-Static"). Meaning the dual bounding method is selected depending on the problem structure at the beginning, in Step 1, and remains fixed throughout the tree. This allows us to work with FP-approximations whenever we do not select primal-bound-shift.

Beyond that, we will analyze whether it is a good idea to compute safe dual bounds only if they are really required, i.e., at nodes where the unsafe bound would lead to pruning ("Auto-Limited"). Furthermore, we experiment with interleaving our actual selection strategy with exact LP solves to eliminate special cases where weak bounds cause the solver to keep branching in subtrees that would have been cut off by the exact LP bound ("Auto-Ileaved").

4 Computational Study

In this section, we investigate the performance of our exact MIP framework employing the different safe dual bounding techniques discussed above: primal-bound-shift ("BoundShift"), project-and-shift ("ProjectShift"), basis verification ("VerifyBasis"), and exact LP solutions ("ExactLP"). We will first look at each method on its own and then examine them within the branch-and-bound algorithm. At the end, we discuss and test strategies to automatically switch between the most promising bounding techniques.

The discussed algorithms were implemented into the branch-and-bound algorithm provided by the MIP framework SCIP 1.2.0.8 [1,2,23], using best bound search for node selection and first fractional variable branching. To solve LPs approximately and exactly we call CPLEX 12.2 [15] and QSOPT_EX 2.5.5 [6], respectively. Rational computations are based on the GMP library 4.3.1 [13]. All benchmark runs were conducted on 2.5 GHz Intel Xeon E5420 CPUs with 4 cores and 16 GB RAM each. To maintain accurate results only one computation was run at the same time. We imposed a time limit of 24 hours and a memory limit of 13 GB. The timings used to measure computation times are always rounded up to one second if they are smaller.

Our test set contains all instances of the MIPLIB 3.0 [8] and MIPLIB 2003 [3] libraries and from the Mittelmann collections [18] that can be solved within 2 hours by the inexact branch-and-bound version of SCIP ("Inexact"). This gives a test suite of 57 MIP instances (`30:70:4_5:0_95:100`, `acc-0`, `acc-1`, `acc-2`, `air03`, `air05`, `bc1`, `bell3a`, `bell5`, `bienst1`, `bienst2`, `blend2`, `dano3_3`, `dano3_4`, `dano3_5`, `dcmulti`, `egout`, `eilD76`, `enigma`, `flugpl`, `gen`, `gesa3`, `gesa3_o`, `irp`, `khb05250`, `l152lav`, `lseu`, `markshare1_1`, `markshare4_0`, `mas76`, `mas284`, `misc03`, `misc07`, `mod008`, `mod010`, `mod011`, `neos5`, `neos8`, `neos11`, `neos21`, `neos897005`, `nug08`, `nw04`, `p0033`, `p0201`, `pk1`, `qap10`, `qnet1_o`, `ran13x13`, `rentacar`, `rgn`, `stein27`, `stein45`, `swath1`, `swath2`, `vpm1`, `vpm2`). Note that we also analyzed the performance on the other, harder, instances of the libraries by looking at the final gap and the number of branch-and-bound nodes processed within a certain time limit. The conclusions drawn here, on the smaller suite, were confirmed by these results.

Table 1. Root node dual bound: Relative difference to "ExactLP" dual bound and additional computation time "DB" in geometric mean

Setting	Zero	S	M	L	∞	DB (s)
BoundShift	13	26	2	0	16	1.0
ProjectShift	19	31	5	0	2	2.8
VerifyBasis	57	0	0	0	0	1.3
ExactLP	57	0	0	0	0	1.4
Auto	20	35	2	0	0	1.3
Auto-Static	21	34	2	0	0	1.3
Auto-Ileaved	20	35	2	0	0	1.3

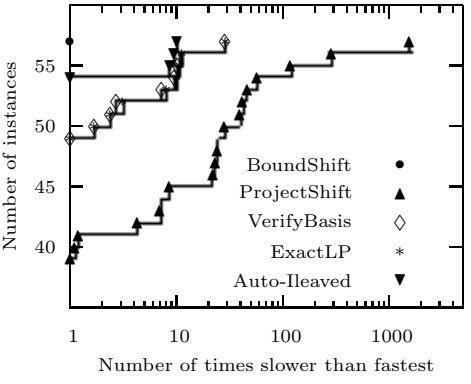


Fig. 1. Comparison of safe dual bounding times "DB" at root

4.1 Root Node Performance

We start with evaluating the root node behavior of the dual bounding methods. Our performance measures are: time overhead and bound quality. The performance profile, see [11], in Fig. 1 visualizes the relative overhead times for the safe dual bounding methods. For each of them, it plots the number of instances for which the safe dual bounding step was performed within a given factor of the bounding time of the fastest method. Table 1 presents the geometric mean of these additional safe dual bounding times in seconds ("DB") and states the number of instances for which a certain dual bound quality was achieved.

This quality is given by the relative difference between the computed safe dual bound $c^\star$ and the exact LP value $c^{\star\star} := \max\{c^T x : x \in LP_j\}$. However, we actually compare the FP-representable upper approximations of both values, as used in Alg. 1, and define the relative difference as $d := (\overline{c^\star} - \overline{c^{\star\star}})/\max\{1, |\overline{c^\star}|, |\overline{c^{\star\star}}|\}$. The corresponding columns in Table 1 are: "Zero" difference for $d = 0$, "S(mall)" difference for $d \in (0, 10^{-9}]$, "M(edium)" difference for $d \in (10^{-9}, 10^{-3}]$, and "L(arge)" difference for $d \in (10^{-3}, \infty)$. Column "∞" counts the worst case behavior, i.e., infinite dual bounds.

We observe that basis verification has similar behavior as exact LP for the root node. However, as we will see in the next section, it will give an improvement over the exact LP solver when expensive basis repair steps are required to find the exact solution.

As expected, primal-bound-shift is the fastest method. However, it produces infinite dual bounds on 16 instances in contrast to only 2 fails for project-and-shift and no fails for basis verification. This is, the bases obtained by CPLEX are often dual feasible and even optimal and project-and-shift meets its requirements most of the time. Still, the finite bounds provided by primal-bound-shift are of very good quality; most of them fall into the "Zero" and "S(mall)" categories. Thus, when primal-bound-shift works we expect to obtain strong bounds and whenever it fails we assume basis verification or project-and-shift to be applicable.

Table 2. Overall performance. "slv" is number of instances solved, "DB" is safe dual bounding time

		Geometric mean for instances solved by all settings (37)		
Setting	slv	Nodes	Time (s)	DB (s)
Inexact	57	18 030	59.4	—
BoundShift	43	24 994	110.4	13.9
ProjectShift	49	18 206	369.3	238.1
VerifyBasis	51	18 078	461.8	329.8
ExactLP	51	18 076	550.7	419.0
Auto	54	18 276	92.6	17.5
Auto-Static	53	18 276	100.2	19.8
Auto-Ileaved	55	18 226	91.4	18.4
Auto-Limited	48	22 035	89.9	12.0

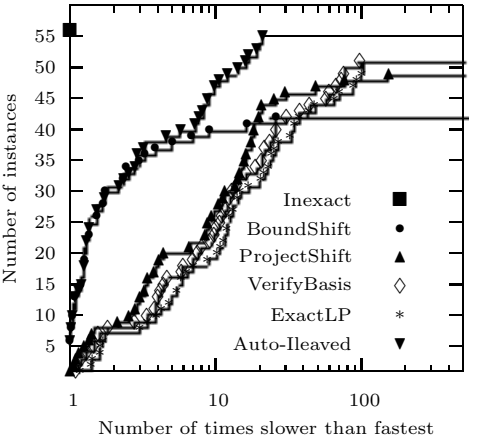


Fig. 2. Comparison of overall solving times "Time"

Where basis verification is in most cases only up to 10 times slower than primal-bound-shift, project-and-shift is up to 100 times slower at the root node because of the expensive initial set-up step. However, as we will see, the overhead incurred by the set-up step of project-and-shift often pays off when it is applied within the entire branch-and-bound tree.

4.2 Overall Performance

We will now analyze the effect of the dual bound methods on the overall performance of the exact MIP solver and compare it with the inexact branch-and-bound version of SCIP ("Inexact"). Table 2, reports the number of instances that were solved within the imposed limits (Column "slv"), for each setting. On 37 instances, all settings succeeded. For this group ("all solved"), we present in Table 2, the number of branch-and-bound nodes "Nodes", the solution time "Time" in seconds, and the additional time spent in the safe dual bounding step "DB" in seconds, all in geometric mean for each method. In addition, Fig. 2 gives a performance profile comparing the solution times. For a setting where an instance had a timeout, it is reported with an infinite ratio to the fastest setting. Thus, the intersection points at the right boarder of the graphic reflect the "slv" column.

The observations made for the root node carry forward to the application in the branch-and-bound algorithm. Primal-bound-shift leads to the fastest node processing. Basis verification has a slightly better performance than solving LPs exactly. However, basis verification is often outperformed by project-and-shift.

A closer look reveals that primal-bound-shift introduces not only a small overhead at the root but throughout the tree as well. For the individual instances it could solve, we usually experience a slow-down of at most 2. The few large

slow-down factors of up to 10 can be explained by a node increase due to a small number of missing variable bounds and by expensive exact LP calls. However, due to its limited applicability we solved only 43 instances.

In contrast, project-and-shift solves 49 instances. That is, the often observed good dual bound quality at the root node seems to stay throughout the tree. We already pointed out the big overhead this costs at the root node. Nevertheless, the method is fast on the rest of the tree and leads to an acceptable slow-down compared to the inexact code. In mean, it is only 6 times slower on the "all solved" group. In this fashion, most of the instances solved by this setting are only up to 20 times slower than the inexact code. If we compare project-and-shift with basis verification we see a similar and often better performance for project-and-shift. Still, on some instances basis verification works better. For example, it solves two more instances of our test set. We examined different problem characteristics and found the number of non-zeros in the constraint matrix to be a good criterion for choosing between project-and-shift and basis verification. In the automatic dual bound selection strategies, we prefer project-and-shift as long as the matrix has at most 10,000 non-zeros.

4.3 Combinations

We already gave some remarks concerning a strategy that automatically chooses a dual bounding method. Another important observation for this purpose is that replacing FP-approximations by FP-relaxations does not affect the performance much: running project-and-shift on an FP-relaxation gave similar results on our test set. Therefore, we decided to always set up an FP-relaxation in Step 1. This way, we are allowed to apply primal-bound-shift at any node we want to.

The automatic decision process used in the "Auto" run works as follows. At every node, we first test whether primal-bound-shift produces a finite bound. If not, we choose project-and-shift or basis verification depending on the structure of the constraint matrix as explained above. The root node results for the combined versions are presented in Table 1 and Fig. 1; the overall performance results can be found in Table 2 and Fig. 2. Note that we excluded "Auto-Limited" from Table 1 as it never computed safe finite bounds at the root node and that we only included the best auto setting in the performance profiles as their graphs look very similar.

The experiments show that "Auto" actually combines the advantages of all dual bounding methods. We can solve all 43 instances that primal-bound-shift solved as well as 11 additional ones by automatically switching to other dual bounding methods at the nodes.

In Sect. 3.5, we discussed three possible improvements for the automatic dual bound selection procedure. The first one, to only guess whether primal-bound-shift will work, is implemented in the test run "Auto-Static". The guess is static, i.e., does not change throughout the tree; we skip primal-bound-shift if more than 20% of the problem variables have lower or upper bounds with absolute

value larger than 10^6. Comparing both automatic settings shows that it is no problem to actually test at each node whether primal-bound-shift works, and it even leads to a slightly improved performance.

The second idea was to interleave the strategy with exact LP calls whenever a node is very likely to be cut off ("Auto-Ileaved"). This does not apply often, but is helpful when it does. We solved one more instance to optimality without introducing a significant time overhead on the other instances. The third extension was to only compute bounds safely if they are actually used for a crucial decision, i.e., if the unsafe bound with tolerances would lead to cutting off a node. Looking at the overall behavior for the corresponding test run "Auto-Limited", it is not clear whether this is a good idea in general. It only solved 48 instead of 54 instances. On the other hand, we experienced that on harder instances the node count at timeout strongly increases, i.e., the node processing is much faster on average. However, we cannot draw any conclusions about the quality of this approach on harder instances as in this setting, the primal-dual-gap does not improve steadily. Further testing is needed here, e.g., by applying additional safe dual bounding steps at selected levels of the tree.

5 Conclusion

From the computational results we can make several key observations. Each dual bounding method studied has strengths and weaknesses depending on the problem structure. Automatically switching between these methods in a smart way solves more problems than any single dual bounding method on its own. Of the 57 problems solved within two hours by the floating-point branch-and-bound solver, 55 could also be solved exactly within 24 hours and the solution time was usually no more than 10 times slower. This demonstrates that the hybrid methodology can lead to an efficient exact branch-and-bound solver, not limited to specific classes of problems. As our focus has been exclusively on the branch-and-bound procedure, we have compared the exact solver against a floating-point solver restricted to pure branch-and-bound. The exact solver is still not directly competitive with the full version of SCIP with its additional features enabled. However, it is realistic to think that the future inclusion of additional MIP machinery such as cutting planes, presolving, and primal heuristics into this exact framework could lead to a full featured exact MIP solver that is not prohibitively slower than its inexact counterparts.

Acknowledgments

The authors would like to thank Tobias Achterberg for helpful discussions on how to best incorporate these features into SCIP. We would also like to thank Daniel Espinoza for his assistance with QSopt_ex, which included adding new features and writing an interface for use within SCIP.

References

1. Achterberg, T.: Constraint Integer Programming. Ph.D. thesis, Technische Universität Berlin (2007)
2. Achterberg, T.: SCIP: Solving constraint integer programs. Math. Programming Computation 1(1), 1–41 (2009)
3. Achterberg, T., Koch, T., Martin, A.: The mixed integer programming library: MIPLIB (2003), http://miplib.zib.de
4. Althaus, E., Dumitriu, D.: Fast and accurate bounds on linear programs. In: Vahrenhold, J. (ed.) SEA 2009. LNCS, vol. 5526, pp. 40–50. Springer, Heidelberg (2009)
5. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2006)
6. Applegate, D.L., Cook, W.J., Dash, S., Espinoza, D.G.: QSopt_ex, http://www.dii.uchile.cl/~daespino/ESolver_doc/main.html
7. Applegate, D.L., Cook, W.J., Dash, S., Espinoza, D.G.: Exact solutions to linear programming problems. Oper. Res. Lett. 35(6), 693–699 (2007)
8. Bixby, R.E., Ceria, S., McZeal, C.M., Savelsbergh, M.W.: An updated mixed integer programming library: MIPLIB 3.0. Optima 58, 12–15 (1998)
9. Cook, W.J., Dash, S., Fukasawa, R., Goycoolea, M.: Numerically safe Gomory mixed-integer cuts. INFORMS J. Comput. 21(4), 641–649 (2009)
10. Dhiflaoui, M., Funke, S., Kwappik, C., Mehlhorn, K., Seel, M., Schömer, E., Schulte, R., Weber, D.: Certifying and repairing solutions to large LPs, how good are LP-solvers? In: SODA 2003, pp. 255–256. ACM/SIAM (2003)
11. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Programming 91(2), 201–213 (2001)
12. Espinoza, D.G.: On Linear Programming, Integer Programming and Cutting Planes. Ph.D. thesis, Georgia Institute of Technology (2006)
13. GMP: GNU multiple precision arithmetic library, http://gmplib.org
14. Goldberg, D.: What every computer scientist should know about floating-point arithmetic. ACM Computing Surveys (CSUR) 23(1), 5–48 (1991)
15. IBM ILOG: CPLEX, http://www.ilog.com/products/cplex
16. Koch, T.: The final NETLIB-LP results. Oper. Res. Lett. 32(2), 138–142 (2004)
17. Kwappik, C.: Exact Linear Programming. Master thesis, Universität des Saarlandes (1998)
18. Mittelmann, H.D.: Benchmarks for Optimization Software (2010), http://plato.asu.edu/bench.html
19. Neumaier, A., Shcherbina, O.: Safe bounds in linear and mixed-integer linear programming. Math. Programming 99(2), 283–296 (2004)
20. Steffy, D.E.: Topics in Exact Precision Mathematical Programming. Ph.D. thesis, Georgia Institute of Technology (2011)
21. de Vries, S., Vohra, R.: Combinatorial Auctions: A Survey. INFORMS J. Comput. 15(3), 284–309 (2003)
22. Wilken, K., Liu, J., Heffernan, M.: Optimal instruction scheduling using integer programming. SIGPLAN Notices 35(5), 121–133 (2000)
23. Zuse Institute Berlin: SCIP, http://scip.zib.de

Valid Inequalities for the Pooling Problem with Binary Variables

Claudia D'Ambrosio[1], Jeff Linderoth[2], and James Luedtke[2]

[1] DEIS, Dipartimento di Elettronica, Informatica e Sistemistica,
Università di Bologna, Italy
c.dambrosio@unibo.it
[2] Department of Industrial and Systems Engineering,
University of Wisconsin-Madison, USA
{linderoth,jrluedt1}@wisc.edu

Abstract. The pooling problem consists of finding the optimal quantity of final products to obtain by blending different compositions of raw materials in pools. Bilinear terms are required to model the quality of products in the pools, making the pooling problem a non-convex continuous optimization problem. In this paper we study a generalization of the standard pooling problem where binary variables are used to model fixed costs associated with using a raw material in a pool. We derive four classes of strong valid inequalities for the problem and demonstrate that the inequalities dominate classic flow cover inequalities. The inequalities can be separated in polynomial time. Computational results are reported that demonstrate the utility of the inequalities when used in a global optimization solver.

Keywords: Integer programming, global optimization, valid inequalities, pooling.

1 Introduction

The pooling problem is to optimize the net cost of products whose composition is determined by blending different raw materials. The blending network consists of three types of nodes: the input streams, representing the raw materials; the pools, where the materials are blended; and the output streams, representing the final products. By deciding the flow quantity passing through the arcs of the this network, the composition of the final products is determined.

There are many applications areas for the pooling problem, including petroleum refining, wastewater treatment, and general chemical engineering process design [1,2,3,4]. Different variants of the pooling problem have been introduced in the literature. In the *standard pooling problem*, the topology of the network is fixed—the pools are fed only by the input streams and connected only to the output streams. The standard pooling problem may be modeled as a non-convex nonlinear program (NLP), where the nonlinearities are bilinear terms that are present to model the (linear) blending process that occurs. The *generalized pooling problem* involves discrete decisions, where the activation of arcs connecting

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 117–129, 2011.

different nodes of the network is to be decided. This problem can be modeled as a non-convex Mixed Integer Nonlinear Program (MINLP). In the *extended pooling problem*, Environmental Protection Agency limits on complex emissions are considered and modeled. In this case, extra variables are introduced and the additional constraints are non-smooth. In many applications of the pooling problem finding the (certified) global optimum can result in significant monetary savings, so a significant research effort has been undertaken on global optimization approaches for the problem. For an overview of the pooling problem variants and the optimization techniques that have been applied successfully for solving such problems, the reader is referred to the recent and exhaustive survey [5].

The focus of this paper is on a special case of the generalized pooling problem, where the topology of a portion of the network is also to be decided. Specifically, there are yes-no decisions associated with connecting the input streams and the pools. For example, this variant of the pooling problem is relevant for crude oil scheduling operations [6,7], where the input streams represent supply ships feeding the storage tanks of a refinery.

Starting from a mathematical formulation of the pooling problem called the PQ-formulation, we extract an interesting set X for which we derive valid inequalities. The set X is a generalization of the well-known single-node fixed-charge flow set [8]. We introduce three classes of inequalities for this set that we call *quality inequalities*, because they are based on the quality target for a final product. A final class of valid inequalities introduced are called *pooling flow cover inequalities*, as they are related to, but dominate, standard flow cover inequalities for the single-node fixed-charge flow set. There are exponentially many pooling flow cover inequalities, and we demonstrate that they can be separated in polynomial time.

The present work is one of the first attempts to generate valid inequalities for a variant of the standard pooling problem that uses specific structures present in the problem. Other approaches have applied general-purpose convexification techniques, such as the reformulation-linearization technique [9] or disjunctive programming [10]. Simultaneously (and independently), Papageorgiou *et al.* [11] take an approach related to ours. Specifically, they study a polyhedral set that serves as a relaxation to a blending problem with fixed-charge structure. The focus of [11] is on developing inequalities for the single product, uncapacitated case, and they are able to find facet-defining inequalities that are useful in computation.

Our new valid inequalities are shown to be quite useful computationally on large instances. Inequalities generated at the root node of a branch-and-bound tree were added to a standard model for the problem and given to the state-of-the-art global optimization solver BARON. With the strengthened model on a test suite of 76 instances solvable by BARON in 2 hours, adding the inequalities reduced BARON's CPU time by a factor 2. An interesting observation from this study is that these useful valid inequalities for this mixed-integer nonconvex set were derived by studying a mixed-integer *linear* relaxation of the set. This

suggests that it may be a useful approach in general to study mixed-integer linear relaxations of global optimization problems.

The remainder of the paper is divided into three sections. Section 2 gives a mathematical description of the problem we study. Section 3 describes the classes of inequalities we have derived, and Section 4 gives the computational results.

2 The Pooling Problem

In this section, we introduce our variant of the standard pooling problem, in which a portion of the topology of the network must be decided. In this variant, a fixed cost is paid if an arc connecting an input stream to a pool is utilized.

2.1 Mathematical Formulation

As a starting point for our model, we use a model of the standard pooling problem that was introduced by Quesada and Grossmann [12]. We use this model because Sahinidis and Tawarmalani [13] have shown that this formulation, which they refer to as *PQ-formulation*, provides a tighter relaxation when the standard McCormick [14] approximation is used to relax the bilinear terms and obtain a Mixed Integer Linear Programming (MILP) problem.

Notation. Input to the pooling problem consists of a network $G = (N, A)$, where the nodes are partitioned into three sets $N = I \cup L \cup J$. The set I is the set of the input streams, the nodes representing the raw materials; the set L is the pool set, where the raw materials are blended; and J is the set of the output streams, the nodes representing the final products. For ease of notation, we assume that there is a complete interconnection network between the nodes of I and L and between the nodes of L and J. That is, $A = \{(i, l) \mid i \in I, l \in L\} \cup \{(l, j) \mid l \in L, j \in J\}$. The inequalities we derive later can easily be generalized to sparse networks. The set K is a set of input and output attributes.

Each input stream $i \in I$ has an associated unit cost c_i and availability A_i. Each output stream $j \in J$ has an associated unit revenue d_j and demand bound D_j. Each pool $l \in L$ has a size capacity S_l. The parameters C_{ik} denote the level of attribute $k \in K$ found in input stream $i \in I$. For each output stream $j \in J$, P^U_{jk} is the upper bound on the composition range of attribute $k \in K$ in the final product j. Finally, there are the parameters f_{il}, which represent the fixed cost that has to be paid if the arc from input stream $i \in I$ to pool $l \in L$ is used.

Decision variables in the formulation are the following:

- q_{il}: proportion of flow from input $i \in I$ to pool $l \in L$;
- y_{lj}: flow from intermediate pool node $l \in L$ to output $j \in J$;
- v_{il}: binary variable with value 1 if the arc from input $i \in I$ to pool $l \in L$ is used, 0 otherwise;
- w_{ilj}: flow from input $i \in I$ to output $j \in J$ through pool $l \in L$.

The PQ-formulation. The objective is to maximize net profit:

$$\min \sum_{j\in J}\left(\sum_{l\in L}\sum_{i\in I} c_i w_{ilj} - \sum_{l\in L} d_j y_{lj}\right) + \sum_{i\in I}\sum_{l\in L} f_{il} v_{il} \ .$$

There are simple constraints that bound raw material availability, pool capacity, and final product demand:

$$\sum_{l\in L}\sum_{j\in J} w_{ilj} \le A_i \quad \forall i \in I; \qquad \sum_{j\in J} y_{lj} \le S_l \quad \forall l \in L; \qquad \sum_{l\in L} y_{lj} \le D_j \quad \forall j \in J \ .$$

A distinguishing feature of the pooling problem is that there is an upper bound on the target value for each attribute of each product:

$$\sum_{l\in L}\sum_{i\in I} C_{ik} w_{ilj} \le P^U_{jk} \sum_{l\in L} y_{lj} \quad \forall j \in J, \forall k \in K \ . \tag{1}$$

The proportion variables q can only be positive if the associated arc was opened:

$$0 \le q_{il} \le v_{il} \quad \forall i \in I, \forall l \in L \ , \tag{2}$$

and must satisfy the properties of proportion:

$$\sum_{i\in I} q_{il} \le 1 \quad \forall l \in L; \qquad \sum_{i\in I} q_{il} \ge v_{i'l} \quad \forall i' \in I, \forall l \in L \ . \tag{3}$$

In the standard pooling problem, without selection of arcs, these constraints would be replaced by $\sum_{i\in I} q_{il} = 1$. In the presence of variable upper bounds (2) on q, the inequalities (3) must be used to allow the possibility that pool l is not used at all, in which case $v_{il} = 0$ for all $i \in I$. If $v_{il} = 1$ for any $i \in I$, then (3) imply the standard equation $\sum_{i\in I} q_{il} = 1$. The w variables are related to the other decision variables as

$$w_{ilj} = q_{il} y_{lj} \quad \forall i \in I, \forall l \in L, \forall j \in J \ . \tag{4}$$

Finally, the PQ-formulation includes the constraints:

$$\sum_{i\in I} w_{ilj} = y_{lj} \quad \forall l \in L, \forall j \in J \ , \tag{5}$$

$$\sum_{j\in J} w_{ilj} \le q_{il} S_l \quad \forall i \in I, \forall l \in L \ . \tag{6}$$

The constraints (6) are redundant, but are added because they can improve the relaxation when the nonconvex constraints (4) are relaxed. Constraints (5) are required to force $y_{lj} = 0$ if pool l is not used (i.e., if $v_{il} = 0$ for all $i \in I$) but are otherwise redundant – and added only to strengthen the relaxation – since then $\sum_{i\in I} q_{il} = 1$ holds and hence (5) follows from (4). This formulation, augmented with appropriate bounds on the variables is given to the global

optimization solver BARON in our subsequent computational experiments. Let $Y_{lj} \stackrel{\text{def}}{=} \min\{S_l, D_j, \sum_{i\in I} A_i\}$ be an upper bound on the flow on from $l \in L$ to $j \in J$. By replacing the nonlinear equations (4) with the well-known McCormick inequalities [14], a linear relaxation of the problem is formed. The McCormick inequalities in this context reduce to the following:

$$0 \le w_{ilj} \le q_{il} Y_{lj}; \quad w_{ilj} \le y_{lj}; \quad w_{ilj} \ge Y_{lj} q_{il} + y_{lj} - Y_{lj} \quad \forall i \in I, \forall l \in L, \forall j \in J\ .$$

Branching on the continuous variables q and y, as well as the binary variables v is required in order to converge to a globally optimal solution.

2.2 Example

To demonstrate the valid inequalities we derive, we will use the simple example shown in Figure 1. There is a single pool, a single product, and a single attribute ($|L| = |J| = |K| = 1$). The three input streams have input quality $C_1 = 3$, $C_2 = 1$, $C_3 = 2$, and there is an upper bound of $P^U = 2.5$ on the target quality. There is an upper bound of $Y = 100$ on the final product.

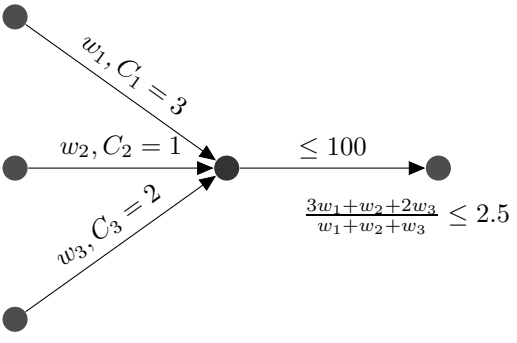


Fig. 1. Pooling Problem Example

3 Valid Inequalities

In this section, we extract an appropriate subset of the constraints of the formulation presented in Section 2.1 and derive a number of strong valid inequalities for this relaxation. To that end, we focus on a single output stream and a single attribute and define the sets

$$I^+ = \{i \in I \mid C_i - P^U \ge 0\}\ , \qquad \text{and } I^- = \{i \in I \mid C_i - P^U < 0\}\ ,$$

where we have dropped the irrelevant indices on the parameters. Next, we define $\alpha_i = |C_i - P^U|\ \forall i \in I$, substitute into equation (1) and use the fact that $y_l = \sum_{i\in I} w_{il}$, to extract the set

$$X = \Big\{(w,q,v) \in \mathbb{R}^{2|I||L|} \times \{0,1\}^{|I||L|} \mid \sum_{l\in L}\sum_{i\in I^+} \alpha_i w_{il} - \sum_{l\in L}\sum_{i\in I^-} \alpha_i w_{il} \le 0;$$
$$\sum_{i\in I} q_{il} \le 1\ \forall l \in L; \quad w_{il} \le Y_l q_{il}, \quad q_{il} \le v_{il}, \quad \forall i \in I, \forall l \in L\Big\}\ ,$$

which is a relaxation of the set of feasible solutions to the PQ-formulation of the generalized pooling problem presented in Section 2.1. The set X is composed of a

single-node flow-type constraint, the definition equations on the proportion variables q, upper bounds on the flow variables based on the proportions q (coming from the McCormick inequalities), and variable upper bounds on the proportion variables q.

3.1 Quality Inequalities

We define the following two classes of inequalities $\forall i \in I^+, \forall i^* \in I^-, \forall l \in L$:

$$\frac{\alpha_1}{\alpha_{i^*}} w_{il} - \sum_{i' \in I^-_>(\alpha_{i^*})} \left(\frac{\alpha_1}{\alpha_{i^*}} - 1 \right) Y_l (v_{i'l} - q_{i'l}) - \frac{\alpha_1}{\alpha_i} Y_l (v_{il} - q_{il}) - \frac{\alpha_1}{\alpha_i \alpha_{i^*}} \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \leq 0 \quad (7)$$

$$\alpha_i w_{il} - \sum_{i' \in I^-_>(\alpha_{i^*})} (\alpha_{i'} - \alpha_{i^*}) w_{i'l} - \alpha_{i^*} Y_l (v_{il} - q_{il}) - \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \leq 0 \quad (8)$$

where $\alpha_1 = \max_{i' \in I^-} \{\alpha_{i'}\}$ and $I^-_>(\alpha_{i^*}) = \{i \in I^- \mid \alpha_i > \alpha_{i^*}\}$.

Proposition 1. *Inequality (7) is valid for X $\forall i \in I^+, \forall i^* \in I^-, \forall l \in L$.*

Proof. Case $v_{il} = 0$: The inequality (7) becomes

$$\sum_{i' \in I^-_>(\alpha_{i^*})} \left(\frac{\alpha_1}{\alpha_{i^*}} - 1 \right) Y_l (v_{i'l} - q_{i'l}) + \frac{\alpha_1}{\alpha_i \alpha_{i^*}} \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \geq 0 \ ,$$

which is valid since all terms are non-negative. In the case $\sum_{i' \in I^-_>(\alpha_{i^*})} v_{i'l} = 0$, inequality (7) reduces to

$$\alpha_i w_{il} \leq \alpha_{i^*} Y_l (1 - q_{il}) + \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'}$$

$$\alpha_i w_{il} \leq \alpha_{i^*} Y_l \Big(\sum_{i' \in I^-} q_{i'l} + \sum_{i' \in I^+ \setminus \{i\}} q_{i'l} \Big) + \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'}$$

$$\alpha_i w_{il} \leq \alpha_{i^*} Y_l \sum_{i' \in I^- \setminus I^-_>(\alpha_{i^*})} q_{i'l} + \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \ , \text{which is always valid.}$$

In the final case $\sum_{i' \in I^-_>(\alpha_{i^*})} v_{i'l} > 0$, we have

$$\frac{\alpha_1}{\alpha_{i^*}} w_{il} \leq \left(\frac{\alpha_1}{\alpha_{i^*}} - 1 \right) Y_l \Big(1 - \sum_{i' \in I^-_>(\alpha_{i^*})} q_{i'l}\Big) + \frac{\alpha_1}{\alpha_i} Y_l (1 - q_{il}) + \frac{\alpha_1}{\alpha_i \alpha_{i^*}} \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'}$$

$$\frac{\alpha_1}{\alpha_{i^*}} w_{il} \leq \left(\frac{\alpha_1}{\alpha_{i^*}} - 1 \right) Y_l q_{il} + \frac{\alpha_1}{\alpha_i} Y_l \sum_{i' \in I^-} q_{i'l} + \frac{\alpha_1}{\alpha_i \alpha_{i^*}} \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'}$$

$$Y_l q_{il} \leq \frac{\alpha_1}{\alpha_i} Y_l \sum_{i' \in I^-} q_{i'l} + \frac{\alpha_1}{\alpha_i \alpha_{i^*}} \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'}$$

$$\alpha_i Y_l q_{il} \leq \alpha_1 Y_l \sum_{i' \in I^-} q_{i'l} + \frac{\alpha_1}{\alpha_{i^*}} \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \ ,$$

which is always valid since it is weaker than the valid inequality

$$\sum_{i' \in I^+} \alpha_{i'} w_{i'l} - \sum_{i' \in I^-} \alpha_{i'} w_{i'l} - \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \leq 0 \ .$$

Proposition 2. *Inequality (8) is valid for* X $\forall i \in I^+, \forall i^* \in I^-, \forall l \in L$.

Proof. In the case $v_{il} = 0$, the inequality (8) reduces to

$$\sum_{i' \in I^-_>(\alpha_{i^*})} (\alpha_{i'} - \alpha_{i^*}) w_{i'l} + \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \geq 0 \ .$$

Otherwise, we have

$$\alpha_i w_{il} - \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \leq \sum_{i' \in I^-_>(\alpha_{i^*})} (\alpha_{i'} - \alpha_{i^*}) w_{i'l} + \alpha_{i^*} Y_l (1 - q_{il})$$

$$\alpha_i w_{il} - \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \leq \sum_{i' \in I^-_>(\alpha_{i^*})} (\alpha_{i'} w_{i'l} + \alpha_{i^*} (Y_l q_{i'l} - w_{i'l})) + \alpha_{i^*} Y_l \sum_{i' \in I^- \setminus I^-_>(\alpha_{i^*})} q_{i'l}$$

$$\alpha_i w_{il} - \sum_{l' \in L \setminus \{l\}} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \leq \sum_{i' \in I^-} \alpha_{i'} w_{i'l} \ .$$

In order to make the definition of inequalities (7) and (8) more clear, consider the example of Section 2.2. For the example, $\alpha_1 = 0.5$, $\alpha_2 = 1.5$, $\alpha_3 = 0.5$ and $I^+ = \{1\}$, $I^- = \{2, 3\}$. The inequalities (7) and (8), defined for indices $i = 1$, $i^* = 3$, respectively, are

$$3w_1 - 200(v_2 - q_2) - 300(v_1 - q_1) \leq 0$$
$$0.5w_1 - w_2 - 50(v_1 - q_1) \leq 0 \ .$$

A final valid inequality limits the flow from inputs that exceed the target quality.

Proposition 3. *The following valid inequality is valid for* X $\forall i \in I^+, \forall l \in L$:

$$\alpha_i w_{il} - Y_l \alpha_1 (v_{il} - q_{il}) - v_{il} \sum_{l' \in L \setminus \{l\}} Y_{l'} \alpha_1 \leq 0 \ . \tag{9}$$

Proof. We consider two cases. Case $v_{il} = 0$: inequality (9) reduces to $0 \leq 0$. Otherwise, using the inequality $q_{il} \leq 1 - \sum_{i' \in I^-} q_{i'l}$ we see that (9) is weaker than $\alpha_i w_{il} - Y_l \alpha_1 \sum_{i' \in I^-} q_{i'l} - \sum_{l' \in L \setminus \{l\}} Y_{l'} \alpha_1 \leq 0$ which is valid because it is weaker than the valid inequality $\sum_{l' \in L} \sum_{i' \in I^+} \alpha_{i'} w_{i'l'} \leq \sum_{l' \in L} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'}$.

Let us consider again the example of Section 2.2. Inequality (9) for $i = 1$ is

$$0.5w_1 - 150(v_1 - q_1) - 0 \leq 0 \ .$$

None of the three classes of inequalities introduced dominates the other, as in general, they can cut off different fractional solutions. Specifically, for the example problem, consider the following three solutions:

1. $q' = (0.1; 0.9; 0)$, $y = 100$, $w' = (10; 90; 0)$, $v' = (0.15; 0.9; 0)$. The inequalities reduce to: $30 - 0 - 15 \leq 0$; $5 - 90 - 2.5 - 0 \leq 0$; $5 - 7.5 \leq 0$ and only the first inequality cuts off this infeasible solution.
2. $q' = (0.5; 0.1; 0.4)$, $y = 100$, $w' = (50; 10; 40)$, $v' = (0.7; 0.9; 0.4)$. The inequalities reduce to: $150 - 160 - 60 \leq 0$; $25 - 10 - 10 \leq 0$; $25 - 30 \leq 0$ and only the second one excludes the fractional solution.
3. $q' = (0.5; 0.5; 0)$, $y = 100$, $w' = (50; 50; 0)$, $v' = (0.6; 1; 0)$. The inequalities reduce to: $150 - 100 - 30 \leq 0$; $25 - 50 - 5 \leq 0$; $25 - 15 \leq 0$ and only the last inequality cuts off the fractional solution.

3.2 Pooling Flow Cover Inequalities

In this section, we focus on the case of a single pool, and hence for notational convenience drop the index l from the discussion. The primary result of the section is the generalization of a flow cover inequality:

Proposition 4. *Let $C^+ \subseteq I^+$ with $\lambda = \sum_{i \in C^+} \alpha_i Y > 0$, $S^- \subseteq I^-$, and define $u^*_{C^+} = \max_{i \in C^+} \alpha_i Y$. The following pooling flow cover inequality is valid for X:*

$$\sum_{i \in C^+} \alpha_i w_i - \sum_{i \in S^-} [u^*_{C^+}(v_i - q_i)] - \sum_{i \in I^- \setminus S^-} \alpha_i w_i \leq 0 \ . \tag{10}$$

Proof. Let $(w, q, v) \in X$ and define the set $T = \{i \mid v_i = 1\}$. If $|S^- \cap T| = 0$, inequality (10) reduces to $\sum_{i \in C^+} \alpha_i w_i - \sum_{i \in I^- \setminus S^-} \alpha_i w_i \leq 0$ which is valid because $w_i = 0$ for all $i \in S^-$. Now suppose $|S^- \cap T| \geq 1$. Since $-\sum_{i \in I^- \setminus S^-} \alpha_i w_i \leq 0$ because $w_i \geq 0 \ \forall i \in I$, it is sufficient to prove that $\sum_{i \in C^+} \alpha_i w_i - \sum_{i \in S^-} [u^*_{C^+}(v_i - q_i)] \leq 0$. First, observe that

$$\begin{aligned} \frac{1}{u^*_{C^+}} \left(\sum_{i \in C^+} \alpha_i w_i - \sum_{i \in S^-} [u^*_{C^+}(v_i - q_i)] \right) &\leq \sum_{i \in C^+} \frac{\alpha_i w_i}{\alpha_i Y} - \sum_{i \in S^-} (v_i - q_i) \\ &\leq \sum_{i \in C^+} q_i - \sum_{i \in S^-} (v_i - q_i) \ . \end{aligned}$$

Thus, we will be done if we prove that $\sum_{i \in C^+} q_i - \sum_{i \in S^-} (v_i - q_i) \leq 0$, which follows from

$$\sum_{i \in C^+} q_i - \sum_{i \in S^- \cap T} (1 - q_i) = \sum_{i \in C^+} q_i + \sum_{i \in S^- \cap T} q_i - |S^- \cap T| \leq 0$$

since $|S^- \cap T| \geq 1$.

The next simple proposition shows that the inequalities (10) are stronger than the generalized flow cover inequalities (see [15]):

$$\sum_{i \in C^+} \alpha_i w_i - \sum_{i \in S^-} \lambda v_i - \sum_{i \in I^- \setminus S^-} \alpha_i w_i \leq 0 \ . \tag{11}$$

Proposition 5. *Inequalities (11) are implied by (10), for every $C^+ \subseteq I^+$ such that $\lambda = \sum_{i \in C^+} \alpha_i Y > 0$ and $S^- \subseteq I^-$.*

Proof. By definition, $u^*_{C^+} \leq \lambda$, and hence

$$-\sum_{i \in S^-} u^*_{C^+}(v_i - q_i) \geq -\sum_{i \in S^-} u^*_{C^+} v_i \geq -\sum_{i \in S^-} \lambda v_i \ .$$

Now let us consider the multiple-pool, one output stream and one attribute case. (Thus, we add index $l \in L$ back to the variables.)

Proposition 6. *For each pool $l \in L$, subset of inputs $S^- \subseteq I^-$, and cover $C^+ \subseteq I^+$, define $u^{*l}_{C^+} = \max_{i \in C^+} \alpha_i Y_l$. The following inequalities are valid for X:*

$$\sum_{i \in C^+} \alpha_i w_{il} - \sum_{i \in S^-} [u^{*l}_{C^+}(v_{il} - q_{il})] - \sum_{i \in I^- \setminus S^-} \alpha_i w_{il} - \sum_{i \in I^-} \sum_{l' \in L \setminus \{l\}} \alpha_i w_{il'} \leq 0 \ . \quad (12)$$

Proof. The proof of 4 is valid also in this case.

The Separation Problem. Given a fractional solution $(w^*, q^*, v^*) \notin X$ we want to find $C^+ \subseteq I^+$, $S^- \subseteq I^-$ and pool $l \in L$ such that (12) is violated. Let the maximum violation of such constraints be

$$z_{\text{SEP}} = \max_{\substack{C^+ \subseteq I^+, \\ l \in L}} \sum_{i \in C^+} \alpha_i w^*_{il} - \sum_{i \in I^-} \min(u^*_{C^+}(v^*_{il} - q^*_{il}), \alpha_i w^*_{il}) - \sum_{i \in I^-} \sum_{l' \in L \setminus \{l\}} \alpha_i w^*_{il'} \ .$$

If $z_{\text{SEP}} > 0$, inequality (12) is violated for (C^+, S^-, l) with $S^- = \{i \in I^- | u^*_{C^+}(v^*_{il} - q^*_{il}) < \alpha_i w^*_{il}\}$. Note that the solution with the maximum violation has the following nice property: if $i \in C^+$, then $i' \in C^+ \ \forall i'$ such that $\alpha_{i'} \leq \alpha_i$. (This follows since, by the definition of $u^*_{C^+}$, if $\alpha_{i'} \leq \alpha_i$, the inequality can only be made more violated by including i' in C^+.) Thus, the separation problem may be solved exactly in polynomial time by considering all the α_i ($i \in I^+$) in non-increasing order over all the pools. Algorithm 1 gives pseudocode for the separation algorithm.

4 Computational Results

The utility of the quality and pooling flow cover inequalities for solving instances of our version of the generalized pooling problem was tested using a cut-and-branch approach. Models were created using the GAMS modeling language both for the original problem and for the continuous linear relaxation of the problem, wherein the nonlinear constraints $w_{ilj} = q_{il} y_{lj}$ are replaced by their McCormick envelopes (as described in Section 2.1) and the constraints $v_{il} \in \{0, 1\}$ are replaced with $0 \leq v_{il} \leq 1$. The continuous relaxation is iteratively solved, where at each iteration, if the solution is fractional and the separation problems find

Algorithm 1. Algorithm for solving the separation problem

set $\hat{\sigma} = -\infty$, $\hat{c} = -\infty$, $\hat{l} = -\infty$;
order α_i such that $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{|I^+|}$ $(i \in I^+)$;
for $c = 1, \ldots, |I^+|$ **do**
  for $l \in L$ **do**
    $\sigma = \sum_{i=1}^{c} \alpha_i w^*_{il} - \sum_{i \in I^-} \min(\alpha_c Y_l (v^*_{il} - q^*_{il}), \alpha_i w^*_{il}) - \sum_{i \in I^-} \sum_{l' \in L \setminus \{l\}} \alpha_i w^*_{il'}$;
    if $\sigma > \hat{\sigma}$ **then**
      set $\hat{\sigma} = \sigma$, $\hat{c} = c$ and $\hat{l} = l$;
    end if
  end for
end for
if $\hat{\sigma} > 0$ **then**
  add cut for (C^+, S^-, l) with $C^+ = \{1, \ldots, \hat{c}\}$, $l = \hat{l}$ and $S^- = \{i \in I^- | u^*_{C^+}(v^*_{il} - q^*_{il}) < \alpha_i w^*_{il}\}$;
end if

violated inequalities, they are added to the linear relaxation, and the relaxation is resolved. The process repeats until the fractional solution can no longer be separated, after which all inequalities generated are added to the MINLP model. The augmented MINLP model is solved with BARON [16] using a CPU time limit of 2 hours. We compare against the performance of BARON on the model without adding the separated quality and pooling flow cover inequalities (7), (8), (9), and (10). Computational results were obtained on a heterogeneous cluster of computers. For each instance, the model was solved on the same machine both with and without cuts, so while the CPU times cannot be compared between instances, the *relative* times between the performance of BARON with and without cuts are comparable.

Our first test suite consisted of 11 instances from the literature, collected by Sahinidis and Tawarmalani [16]. For these instances, a fixed cost of 500 was added to every input arc. Some of these instances contain bypass arcs, or arcs that connect input streams directly to outputs are present. Bypass arcs are treated as additional pools with only one input in the inequalities. Results of the experiment are given in Table 1, where we report for each instance, the value of the global optimum, the number of nodes and the CPU time needed for BARON to find the optimum both with and without the addition of the violated quality and pooling flow cover inequalities, and the number of inequalities found. In general, the instances from the literature were far too small to draw any meaningful conclusions. The number of nodes is fewer in 4 of 11 cases, in 6 cases it is the same, and in one case, more nodes are taken after adding the inequalities.

A second test set consisted of 90 randomly generated instances of various sizes. The random instances were parameterized by the average graph density β, the number of inputs $|I|$, the number of pools $|L|$, the number of outputs $|J|$, and the number of attributes $|K|$, and named $\beta - |I| - |L| - |J| - |K|$ based on their attributes. For each combination of tested parameters, 10 instances were generated, and we report average performance results over all instances in the family.

Table 1. BARON Performance With and Without Cutting Planes

		no cuts			with cuts	
	GO	# nodes	CPU time	# cuts	# nodes	CPU time
adhya1	630.0	7	0.14	20	7	0.14
adhya2	630.0	1	0.07	15	5	0.11
adhya3	978.0	17	0.43	8	1	0.12
adhya4	529.2	17	0.84	12	7	0.54
bental4	500.0	9	0.02	4	1	0.02
bental5	-2000.0	1	0.12	0	1	0.13
foulds2	266.7	1	0.04	6	1	0.02
haverly1	500.0	9	0.01	4	1	0.01
haverly2	400.0	9	0.01	4	9	0.01
haverly3	100.0	1	0.01	4	1	0.01
rt2	-1672.6	356	1.02	0	356	1.05

Table 2. Average Performance of BARON With and Without Cutting Planes

		No Cuts			With Cuts	
Instance Family	# Solved	Nodes	Time	# Cuts	Nodes	Time
20-15-10-10-1	9	1052	14.9	11.1	383	7.3
20-15-10-10-2	10	7850	638.2	15.3	3338	440.3
20-15-10-10-4	10	2637	109.5	11.9	2241	168.5
30-15-5-10-1	9	22009	520.5	12.8	13095	367.4
30-15-5-10-2	8	7384	406.8	19.6	3988	239.0
30-15-5-10-4	10	6041	361.1	27.0	1884	109.9
30-15-8-10-1	3	21971	1689.6	11.7	6504	478.3
30-15-8-10-2	9	15663	823.9	19.7	4303	337.6
30-15-8-10-4	8	30424	1035.3	22.0	5472	457.2

All the instances are available at the webpage `http://www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm`. Results of this experiment are summarized in Table 2.

Without adding inequalities, BARON is able to solve 76 of the 90 instances. Adding the inequalities, BARON is able to solve three additional instances of the 90. Columns "Nodes", "Time", and "# Cuts" have to be intended as average values. Averages in Table 2 are taken only over the instances that both methods can solve. Complete results detailed the computational performance on specific instances are given on the web site `http://www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm`.

For the 76 solved instances, the total CPU time required to solve the instances reduces from 39927 seconds to 20602 seconds when the cuts are added to the model before calling BARON. The total number of nodes required to solve the 76 instances reduces from 882173 to 329851 by adding cuts. The most significant performance improvement seems to occur on the instances that have 30%

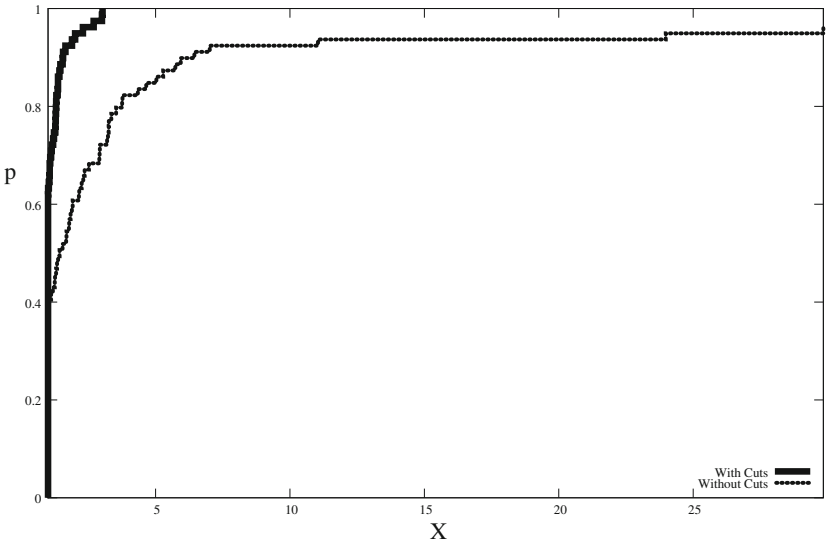


Fig. 2. Performance Profile of Solution Time

network density and 8 pools (in the `30-15-8-10-x` family), indicating that larger, denser instances may benefit most from the additional of the quality and pooling flow cover inequalities.

A performance profile of the CPU time of the 79 instances solvable by either method is given in Figure 2. In the curves shown in Figure 2, the point (X, p) is on the graph for the associated method if a fraction p of the 79 instances solved by either method were solved within a factor X of the best of the two methods. For a more complete description of performance profiles, the reader is referred to [17]. From the results of the second experiment, it is clear that the addition of the inequalities has a significant beneficial impact on computational performance.

Acknowledgments. The authors would like to thank Ahmet Keha for bringing reference [11] to their attention. This work benefitted from numerous insightful discussions with Andrew Miller. This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S Department of Energy, under Grants DE-FG02-08ER25861 and DE-FG02-09ER25869.

References

1. DeWitt, C., Lasdon, L., Waren, A., Brenner, D., Melham, S.: OMEGA: An improved gasoline blending system for Texaco. Interfaces 19, 85–101 (1989)
2. Rigby, B., Lasdon, L., Waren, A.: The evolution of Texaco's blending systems: From OMEGA to StarBlend. Interfaces 25, 64–83 (1995)
3. Bagajewicz, M.: A review of recent design procedures for water networks in refineries and process plants. Computers & Chemical Engineering 24, 2093–2113 (2000)

4. Kallrath, J.: Mixed integer optimization in the chemical process industry: Experience, potential and future perspectives. Chemical Engineering Research and Design 78, 809–822 (2000)
5. Misener, R., Floudas, C.: Advances for the pooling problem: Modeling, global optimization, & computational studies. Applied and Computational Mathematics 8, 3–22 (2009)
6. Lee, H., Pinto, J., Grossmann, I., Park, S.: Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. Industrial & Engineering Chemistry Research 35, 1630–1641 (1996)
7. Shah, N.: Mathematical programming techniques for crude oil scheduling. Computers & Chemical Engineering 20, S1227–S1232 (1996)
8. Padberg, M., Van Roy, T.J., Wolsey, L.: Valid linear inequalities for fixed charge problems. Operations Research 33, 842–861 (1985)
9. Meyer, C., Floudas, C.: Global optimization of a combinatorially complex generalized pooling problem. AIChE Journal 52, 1027–1037 (2006)
10. Karuppiah, R., Grossmann, I.: Global optimization for the synthesis of integrated water systems in chemical processes. Computers & Chemical Engineering 30, 650–673 (2006)
11. Papageorgiou, D.J., Toriello, A., Nemhauser, G.L., Savelsbergh, M.: Fixed-charge transportation with product blending (2010) (unpublished manuscript)
12. Quesada, I., Grossmann, I.: Global optimization of bilinear process networks with multicomponent flows. Computers and Chemical Engineering 19, 1219–1242 (1995)
13. Tawarmalani, M., Sahinidis, N.: Convexification and global optimization in continuous and mixed-integer nonlinear programming: Theory, Algorithms, Software, and Applications. Kluwer Academic Publishers, Dordrecht (2002)
14. McCormick, G.: Computability of global solutions to factorable nonconvex programs: Part 1 - convex underestimating problems. Mathematical Programming 10, 147–175 (1976)
15. Wolsey, L., Nemhauser, G.: Integer and Combinatorial Optimization. Wiley, New York (1988)
16. Sahinidis, N., Tawarmalani, M.: Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints. Journal of Global Optimization 32, 259–280 (2005)
17. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Mathematical Programming 91, 201–213 (2002)

On the Chvátal-Gomory Closure of a Compact Convex Set

Daniel Dadush[1], Santanu S. Dey[1], and Juan Pablo Vielma[2]

[1] H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
dndadush@gatech.edu, santanu.dey@isye.gatech.edu
[2] Department of Industrial Engineering, University of Pittsburgh, 1048 Benedum Hall, Pittsburgh, PA 15261, USA
jvielma@pitt.edu

Abstract. In this paper, we show that the Chvátal-Gomory closure of any compact convex set is a rational polytope. This resolves an open question of Schrijver [15] for irrational polytopes[1], and generalizes the same result for the case of rational polytopes [15], rational ellipsoids [7] and strictly convex bodies [6].

Keywords: Chvátal-Gomory Closure, Compact Sets.

1 Introduction

Gomory [11] introduced the Gomory fractional cuts, also known as Chvátal-Gomory (CG) cuts [5], to design the first finite cutting plane algorithm for Integer Linear Programming (ILP). Since then, many important classes of facet-defining inequalities for combinatorial optimization problems have been identified as CG cuts. For example, the classical Blossom inequalities for general Matching [9] - which yield the integer hull - and Comb inequalities for the Traveling Salesman problem [12,13] are both CG cuts over the base linear programming relaxations. CG cuts have also been effective from a computational perspective; see for example [2,10]. Although CG cuts have traditionally been defined with respect to rational polyhedra for ILP, they straightforwardly generalize to the nonlinear setting and hence can also be used for convex Integer Nonlinear Programming (INLP), i.e. the class of discrete optimization problems whose continuous relaxation is a general convex optimization problem. CG cuts for non-polyhedral sets were considered implicitly in [5,15] and more explicitly in [4,6,7]. Let $K \subseteq \mathbb{R}^n$ be a closed convex set and let h_K represent its support function, i.e. $h_K(a) = \sup\{\langle a, x\rangle : x \in K\}$. Given $a \in \mathbb{Z}^n$, we define the CG cut for K derived from a as the inequality

$$\langle a, x\rangle \leq \lfloor h_K(a) \rfloor \ . \tag{1}$$

[1] After the completion of this work, it has been brought to our notice that the polyhedrality of the Chvátal-Gomory Closure for irrational polytopes has recently been shown independently by J. Dunkel and A. S. Schulz in [8]. The proof presented in this paper has been obtained independently.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 130–142, 2011.

The CG closure of K is the convex set whose defining inequalities are exactly all the CG cuts for K. A classical result of Schrijver [15] is that the CG closure of a rational polyhedron is a rational polyhedron. Recently, we were able to verify that the CG closure of any strictly convex body[2] intersected with a rational polyhedron is a rational polyhedron [7,6]. We remark that the proof requires techniques significantly different from those described in [15].

While the intersections of strictly convex bodies with rational polyhedra yield a large and interesting class of bodies, they do not capture many natural examples that arise in convex INLP. For example, it is not unusual for the feasible region of a semi-definite or conic-quadratic program [1] to have infinitely many faces of different dimensions, where additionally a majority of these faces cannot be isolated by intersecting the feasible region with a rational supporting hyperplane (as is the case for standard ILP with rational data). Roughly speaking, the main barrier to progress in the general setting has been a lack of understanding of how CG cuts act on irrational affine subspaces (affine subspaces whose defining equations cannot be described with rational data).

As a starting point for this study, perhaps the simplest class of bodies where current techniques break down are polytopes defined by irrational data. Schrijver considers these bodies in [15], and in a discussion section at the end of the paper, he writes [3]:

> "We do not know whether the analogue of Theorem 1 is true in real spaces. We were able to show only that if P is a bounded polyhedron in real space, and P' has empty intersection with the boundary of P, then P' is a (rational) polyhedron."

In this paper, we prove that the CG closure of any compact convex set[4] is a rational polytope, thus also resolving the question raised in [15]. As seen by Schrijver [15], most of the "action" in building the CG closure will indeed take place on the boundary of K. While the proof presented in this paper has some high level similarities to the one in [6], a substantially more careful approach was required to handle the general facial structure of a compact convex set (potentially infinitely many faces of all dimensions) and completely new ideas were needed to deal with faces having irrational affine hulls (including the whole body itself).

This paper is organized as follows. In Section 2 we introduce some notation, formally state our main result and give an overview of the proof. We then proceed with the full proof which is presented in Sections 3-5. In Section 6, we present

[2] A full dimensional compact convex set whose only non-trivial faces are vertices, i.e. of dimension 0.

[3] Theorem 1 in [15] is the result that the CG closure is a polyhedron. P' is the notation used for CG closure in [15].

[4] If the convex hull of integer points in a convex set is not polyhedral, then the CG closure cannot be expected to be polyhedral. Since we do not have a good understanding of when this holds for unbounded convex sets, we restrict our attention here to the CG closure of compact convex sets.

a generalization of Integer Farkas' Lemma that is a consequence of the proof techniques developed in this paper.

2 Definitions, Main Result and Proof Idea

Definition 1 (CG Closure). *For a convex set $K \subseteq \mathbb{R}^n$ and $S \subseteq \mathbb{Z}^n$ let $CC(K,S) := \bigcap_{a \in S}\{x \in \mathbb{R}^n : \langle x, y\rangle \leq \lfloor h_K(y)\rfloor\}$. The CG closure of K is defined to be the set $CC(K) := CC(K, \mathbb{Z}^n)$.*

The following theorem is the main result of this paper.

Theorem 1. *If $K \subseteq \mathbb{R}^n$ is a non-empty compact convex set, then $CC(K)$ is finitely generated. That is, there exists $S \subseteq \mathbb{Z}^n$ such that $|S| < \infty$ and $CC(K) = CC(K,S)$. In particular $CC(K)$ is a rational polyhedron.*

We will use the following definitions and notation: For $x, y \in \mathbb{R}^n$, let $[x,y] = \{\lambda x + (1-\lambda)y : 0 \leq \lambda \leq 1\}$ and $(x,y) = [x,y] \setminus \{x,y\}$. Let $B^n := \{x \in \mathbb{R}^n : \|x\| \leq 1\}$ and $S^{n-1} := \mathrm{bd}(B^n)$. (bd stands for boundary.) For a convex set K and $v \in \mathbb{R}^n$, let $H_v(K) := \{x \in \mathbb{R}^n : \langle v, x\rangle \leq h_K(v)\}$ denote the supporting halfspace defined by v for K, and let $H_v^=(K) := \{x \in \mathbb{R}^n : \langle v, x\rangle = h_K(v)\}$ denote the supporting hyperplane. $F \subseteq K$ is a face of K if for every line segment $[x,y] \subseteq K$, $[x,y] \cap F \neq \emptyset \Rightarrow [x,y] \subseteq F$. A face F of K is proper if $F \neq K$. Let $F_v(K) := K \cap H_v^=(K)$ denote the face of K exposed by v. If the context is clear, then we drop the K and simply write H_v, $H_v^=$ and F_v. For $A \subseteq \mathbb{R}^n$, let $\mathrm{aff}(A)$ denote the smallest affine subspace containing A. Furthermore let $\mathrm{aff}_I(A) := \mathrm{aff}(\mathrm{aff}(A) \cap \mathbb{Z}^n)$, i.e. the largest integer subspace in $\mathrm{aff}(A)$.

We present the outline of the proof for Theorem 1. The proof proceeds by induction on the dimension of K. The base case (K is a single point) is trivial. By the induction hypothesis, we can assume that (†) every proper exposed face of K has a finitely generated CG closure. We build the CG closure of K in stages, proceeding as follows:

1. (Section 3) For F_v, a proper exposed face, where $v \in \mathbb{R}^n$, show that $\exists\, S \subseteq \mathbb{Z}^n$, $|S| < \infty$ such that $CC(K,S) \cap H_v^= = CC(F_v)$ and $CC(K,S) \subseteq H_v$ using (†) and by proving the following:
   (a) (Section 3.1) A CG cut for F_v can be rotated or "lifted" to a CG cut for K such that points in $F_v \cap \mathrm{aff}_I(H_v^=)$ separated by the original CG cut for F_v are separated by the new "lifted" one.
   (b) (Section 3.2) A finite number of CG cuts for K separate all points in $F_v \setminus \mathrm{aff}_I(H_v^=)$ and all points in $\mathbb{R}^n \setminus H_v$.
2. (Section 4) Create an approximation $CC(K,S)$ of $CC(K)$ such that (i) $|S| < \infty$, (ii) $CC(K,S) \subseteq K \cap \mathrm{aff}_I(K)$ (iii) $CC(K,S) \cap \mathrm{relbd}(K) = CC(K) \cap \mathrm{relbd}(K)$. This is done in two steps:
   (a) (Section 4.1) Using the lifted CG closures of F_v from 1. and a compactness argument on the sphere, create a first approximation $CC(K,S)$ satisfying (i) and (ii).

(b) (Section 4.2) Noting that $CC(K,S) \cap \text{relbd}(K)$ is contained in the union of a finite number of proper exposed faces of K, add the lifted CG closures for each such face to S to satisfy (iii).

3. (Section 5) We establish the final result by showing that there are only a finite number of CG cuts which separate a least one vertex of the approximation of the CG closure from (2).

3 $CC(K,S) \cap H_v^= = CC(F_v)$ and $CC(K,S) \subseteq H_v$

When K is a rational polyhedron, a key property of the CG closure is that for every face F of K, we have that $(*)$ $CC(F) = F \cap CC(K)$. In this setting, a relatively straightforward induction argument coupled with $(*)$ allows one to construct the approximation of the CG closure described above. In our setting, where K is compact convex, the approach taken is similar in spirit, though we will encounter significant difficulties. First, since K can have infinitely many faces, we must couple our induction with a careful compactness argument. Second and more significantly, establishing $(*)$ for compact convex sets is substantially more involved than for rational polyhedra. As we will see in the following sections, the standard lifting argument to prove $(*)$ for rational polyhedra cannot be used directly and must be replaced by a more involved two stage argument.

3.1 Lifting CG Cuts

To prove $CC(F) = F \cap CC(K)$ one generally uses a 'lifting approach', i.e., given a CG cut $CC(F,\{w\})$ for F, $w \in \mathbb{Z}^n$, we show that there exists a CG cut $CC(K,\{w'\})$ for K, $w' \in \mathbb{Z}^n$, such that

$$CC(K,\{w'\}) \cap \text{aff}(F) \subseteq CC(F,\{w\}) \cap \text{aff}(F). \tag{2}$$

To prove (2) when K is a rational polyhedron, one proceeds as follows. For the face F of K, we compute $v \in \mathbb{Z}^n$ such that $F_v(K) = F$ and $h_K(v) \in \mathbb{Z}$. For $w \in \mathbb{Z}^n$, we return the lifting $w' = w + lv$, $l \in \mathbb{Z}_{>0}$, where l is chosen such that $h_K(w') = h_F(w')$. For general convex bodies though, neither of these steps may be achievable. When K is strictly convex however, in [6] we show that the above procedure can be generalized. First, every proper face F of K is an exposed vertex, hence $\exists\ x \in K, v \in \mathbb{R}^n$ such that $F = F_v = \{x\}$. For $w \in \mathbb{Z}^n$, we show that setting $w' = w + v'$, where v' is a fine enough Dirichlet approximation (see Theorem 2 below) to a scaling of v is sufficient for (2). In the proof, we critically use that F is simply a vertex. In the general setting, when K is a compact convex set, we can still meaningfully lift CG cuts, but not from all faces and not with exact containment. First, we only guarantee lifting for an exposed face F_v of K. Second, when lifting a CG cut for F_v derived from $w \in \mathbb{Z}^n$, we only guarantee the containment on $\text{aff}_I(H_v^=)$, i.e. $CC(K,w') \cap \text{aff}_I(H_v^=) \subseteq CC(F,w) \cap \text{aff}_I(H_v^=)$. This lifting, Proposition 1 below, uses the same Dirichlet approximation technique as

in [6] but with a more careful analysis. Since we only guarantee the behavior of the lifting w' on $\text{aff}_I(H_v^=)$, we will have to deal with the points in $\text{aff}(F)\backslash\text{aff}_I(H_v^=)$ separately, which we discuss in the next section.

The next lemma describes the central mechanics of the lifting process explained above. The sequence $(w_i)_{i=1}^\infty$ will eventually denote the sequence of Dirichlet approximates of the scaling of v added to w, where one of these will serve as the lifting w'. We skip the proof due to lack of space.

Lemma 1. *Let $K \subseteq \mathbb{R}^n$ be a compact convex set. Take $v, w \in \mathbb{R}^n$, $v \neq 0$. Let $(w_i, t_i)_{i=1}^\infty$, $w_i \in \mathbb{R}^n, t_i \in \mathbb{R}_+$ be a sequence such that*

$$a.\ \lim_{i\to\infty} t_i = \infty, \quad b.\ \lim_{i\to\infty} w_i - t_i v = w. \tag{3}$$

Then for every $\epsilon > 0$ there exists $N_\epsilon \geq 0$ such that for all $i \geq N_\epsilon$

$$h_K(w_i) + \epsilon \geq t_i h_K(v) + h_{F_v(K)}(w) \geq h_K(w_i) - \epsilon. \tag{4}$$

Theorem 2 (Dirichlet's Approximation Theorem). *Let $(\alpha_1, \ldots, \alpha_l) \in \mathbb{R}^l$. Then for every positive integer N, there exists $1 \leq n \leq N$ such that $\max_{1\leq i\leq l} |n\alpha_i - \lfloor n\alpha_i \rceil| \leq 1/N^{1/l}$.*

Proposition 1. *Let $K \subseteq \mathbb{R}^n$ be a compact and convex set, $v \in \mathbb{R}^n$ and $w \in \mathbb{Z}^n$. Then $\exists w' \in \mathbb{Z}^n$ such that $CC(K, w') \cap \text{aff}_I(H_v^=(K)) \subseteq CC(K, w) \cap \text{aff}_I(H_v^=(K))$.*

Proof. First, by possibly multiplying v by a positive scalar we may assume that $h_K(v) \in \mathbb{Z}$. Let $S = \text{aff}_I(H_v^=(K))$. We may assume that $S \neq \emptyset$, since otherwise the statement is trivially true.

From Theorem 2 for any $v \in \mathbb{R}^n$ there exists $(s_i, t_i)_{i=1}^\infty$, $s_i \in \mathbb{Z}^n$, $t_i \in \mathbb{N}$ such that (a.) $t_i \to \infty$ and (b.) $\|s_i - t_i v\| \to 0$. Now define the sequence $(w_i, t_i)_{i=1}^\infty$, where $w_i = w + s_i$, $i \geq 1$. Note that the sequence (w_i, t_i) satisfies (3) and hence by Lemma 1 for any $\epsilon > 0$, there exists N_ϵ such that (4) holds. Let $\epsilon = \frac{1}{2}\big(1 - (h_{F_v(K)}(w) - \lfloor h_{F_v(K)}(w) \rfloor)\big)$, and let $N_1 = N_\epsilon$. Note that $\lfloor h_{F_v(K)}(w) + \epsilon \rfloor = \lfloor h_{F_v(K)}(w) \rfloor$. Hence, since $h_K(v) \in \mathbb{Z}$ by assumption, for all $i \geq N_1$ we have that $\lfloor h_K(w_i) \rfloor \leq \lfloor t_i h_K(v) + h_{F_v(K)}(w) + \epsilon \rfloor = t_i h_K(v) + \lfloor h_{F_v(K)}(w) + \epsilon \rfloor = t_i h_K(v) + \lfloor h_{F_v(K)}(w) \rfloor$.

Now pick $z_1, \ldots, z_k \in S \cap \mathbb{Z}^n$ such that $\text{aff}(z_1, \ldots, z_k) = S$ and let $R = \max\{\|z_j\| : 1 \leq j \leq k\}$. Choose N_2 such that $\|w_i - t_i v - w\| \leq \frac{1}{2R}$ for $i \geq N_2$. Now note that for $i \geq N_2$, $|\langle z_j, w_i\rangle - \langle z_j, t_i v + w\rangle| = |\langle z_j, w_i - t_i v - w\rangle| \leq \|z_j\|\|w_i - t_i v - w\| \leq R\frac{1}{2R} = \frac{1}{2} \quad \forall j \in \{1, \ldots, k\}$.

Next note that since $z_j, w_i \in \mathbb{Z}^n$, $\langle z_j, w_i\rangle \in \mathbb{Z}$. Furthermore, $t_i \in \mathbb{N}$, $\langle v, z_j\rangle = h_K(v) \in \mathbb{Z}$ and $w \in \mathbb{Z}^n$ implies that $\langle z_j, t_i v + w\rangle \in \mathbb{Z}$. Given this, we must have $\langle z_j, w_i\rangle = \langle z_j, t_i v + w\rangle \quad \forall j \in \{1, \ldots, k\}$, $i \geq 1$ and hence we get $\langle x, w_i\rangle = \langle x, t_i v + w\rangle \quad \forall x \in S$, $i \geq 1$.

Let $w' = w_i$ where $i = \max\{N_1, N_2\}$. Now examine the set $L = \{x : \langle x, w'\rangle \leq \lfloor h_K(w')\rfloor\} \cap S$. Here we get that $\langle x, w_i\rangle \leq t_i h_K(v) + \lfloor h_{F_v(K)}(w)\rfloor$ and $\langle x, v\rangle = h_K(v)$ for all $x \in L$ Hence, we see that $\langle x, w_i - t_i v\rangle \leq \lfloor h_{F_v(K)}(w)\rfloor$ for all $x \in L$. Furthermore, since $\langle x, w_i - t_i v\rangle = \langle x, w\rangle$ for all $x \in L \subseteq S$, we have that $\langle x, w\rangle \leq \lfloor h_{F_v(K)}(w)\rfloor$ for all $x \in L$, as needed.

3.2 Separating All Points in $F_v \setminus \text{aff}_I(H_v^=)$

Since the guarantees on the lifted CG cuts produced in the previous section are restricted to $\text{aff}_I(H_v^=)$, we must still deal with the points in $F_v \setminus \text{aff}_I(H_v^=)$. In this section, we show that points in $F_v \setminus \text{aff}_I(H_v^=)$ can be separated by using a finite number of CG cuts in Proposition 2. To prove this, we will need Kronecker's theorem on simultaneous diophantine approximation which is stated next. See Niven [14] or Cassels [3] for a proof.

Theorem 3. *Let $(x_1, \ldots, x_n) \in \mathbb{R}^n$ be such that the numbers $x_1, \ldots, x_n, 1$ are linearly independent over $\mathbb{Q}$. Then the set $\{(nx_1 \pmod 1), \ldots, nx_n \pmod 1)) : n \in \mathbb{N}\}$ is dense in $[0,1)^n$.*

We state the following lemmas without proof which allow us to normalize vector v defining F_v and $H_v^=$ and simplify the analysis that follows.

Lemma 2. *Let $K \subseteq \mathbb{R}^n$ be a closed convex set, and let $T : \mathbb{R}^n \to \mathbb{R}^n$ be an invertible linear transformation. Then $h_K(v) = h_{TK}(T^{-t}v)$ and $F_v(K) = T^{-1}(F_{T^{-t}v}(TK))$ for all $v \in \mathbb{R}^n$. Furthermore, if T is a unimodular transformation, then $CC(K) = T^{-1}(CC(TK))$.*

Lemma 3. *Take $v \in \mathbb{R}^n$. Then there exists an unimodular transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ and $\lambda \in \mathbb{Q}_{>0}$ such that for $v' = \lambda Tv$ we get that*

$$v' = \left(\underbrace{0, \ldots, 0}_{t \text{ times}}, \underbrace{1}_{s \text{ times}}, \alpha_1, \ldots, \alpha_r \right), \tag{5}$$

where $t, r \in \mathbb{Z}_+$, $s \in \{0,1\}$, and $\{1, \alpha_1, \ldots, \alpha_r\}$ are linearly independent over $\mathbb{Q}$. Furthermore, we have that $\mathcal{D}(v) = \inf\{\dim(W) : v \in W, W = \{x \in \mathbb{R}^n : Ax = 0\}, A \in \mathbb{Q}^{m \times n}\} = s + r$.

We now show that the points in $F_v \setminus \text{aff}_I(H_v^=)$ can be separated using a finite number of CG cuts. We first give a rough sketch of the proof. We restrict to the case where $\text{aff}_I(H_v^=) \neq \emptyset$. From here one can verify that any rational affine subspace contained in $\text{aff}(H_v^=)$ must also lie in $\text{aff}_I(H_v^=)$. Next we use Kronecker's theorem to build a finite set $C \subseteq \mathbb{Z}^n$, where each vector in C is at distance at most ϵ from some scaling of v, and where v can be expressed as a non-negative combination of the vectors in C. By choosing ϵ and the scalings of v appropriately, we can ensure that the CG cuts derived from C dominate the inequality $\langle v, x \rangle \leq h_K(v)$, i.e. $CC(K, C) \subseteq H_v$. If $CC(K, C)$ lies in the interior of $H_v(K)$, we have separated all of $H_v^=$ (including $F_v \setminus \text{aff}_I(H_v^=)$) and hence are done. Otherwise, $T := CC(K, C) \cap H_v^=$ is a face of a rational polyhedron, and therefore $\text{aff}(T)$ is a rational affine subspace. Since $\text{aff}(T) \subseteq \text{aff}(H_v^=)$, as discussed above $T \subseteq \text{aff}(T) \subseteq \text{aff}_I(H_v^=)$ as required.

Proposition 2. *Let $K \subseteq \mathbb{R}^n$ be a compact convex set and $v \in \mathbb{R}^n$. Then there exists $C \subseteq \mathbb{Z}^n$, $|C| \leq \mathcal{D}(v) + 1$, such that*

$$CC(K, C) \subseteq H_v(K) \quad \text{and} \quad CC(K, C) \cap H_v^=(K) \subseteq \text{aff}_I(H_v^=(K)).$$

Proof. By scaling v by a positive scalar if necessary, we may assume that $h_K(v) \in \{0, 1, -1\}$. Let T and λ denote the transformation and scaling promised for v in Lemma 3. Note that $T^{-t}\{x \in \mathbb{R}^n : \langle v, x\rangle = h_K(v)\} = \{x \in \mathbb{R}^n : \langle v, T^t x\rangle = h_K(v)\} = \{x \in \mathbb{R}^n : \langle \lambda T v, x\rangle = h_{T^{-t}K}(\lambda T v)\}$.

Now let $v' = \lambda T v$ and $b' = h_{T^{-t}K}(\lambda T v)$. By Lemma 2, it suffices to prove the statement for v' and $K' = T^{-t}K$. Now v' has the form (5) where $t, r \in \mathbb{Z}_+$, $s \in \{0, 1\}$, and $(1, \alpha_1, \ldots, \alpha_r)$ are linearly independent over $\mathbb{Q}$. For convenience, let $k = s + t$, where we note that $v'_{k+1}, \ldots, v'_{k+r} = (\alpha_1, \ldots, \alpha_r)$.

Claim 1: Let $S = \{x \in \mathbb{Z}^n : \langle v', x\rangle = b'\}$. Then S satisfies one of the following: (1) $S = \mathbb{Z}^t \times b' \times 0^r$: $s = 1, b' \in \mathbb{Z}$, (2) $S = \mathbb{Z}^t \times 0^r$: $s = 0, b' = 0$, (3) $S = \emptyset$: $s = 0, b' \neq 0$ or $s = 1, b' \notin \mathbb{Z}$.

Claim 2: Let $I = \{nv' \pmod 1 : n \in N\}$. Then Theorem 3 implies that I is dense in $0^k \times [0, 1)^r$.

Due to space restriction, we skip the proofs of these two claims and from now on we only consider the case where $S \neq \emptyset$.

Claim 3: There exists $a_1, \ldots, a_{r+1} \subseteq \mathbb{Z}^n$ and $\lambda_1, \ldots, \lambda_{r+1} \geq 0$ such that $\sum_{i=1}^{r+1} \lambda_i a_i = v'$ and $\sum_{i=1}^{r+1} \lambda_i \lfloor h'_K(a_i)\rfloor \leq b'$.

Since K' is compact, there exists $R > 0$ such that $K' \subseteq RB^n$. Take the subspace $W = 0^k \times \mathbb{R}^r$. Let $w_1, \ldots, w_{r+1} \in W \cap S^{n-1}$, be any vectors such that for some $0 < \epsilon < 1$ we have $\sup_{1 \leq i \leq r+1} \langle w_i, d\rangle \geq \epsilon$ for all $d \in S^{n-1} \cap W$ (e.g. $w_1, \ldots, w_{r+1}$ are the vertices of a scaled isotropic r-dimensional simplex). Let $a = \frac{1}{8}\min\{\frac{1}{R}, \epsilon\}$, and $b = \frac{1}{2}\epsilon a$. Now, for $1 \leq i \leq r+1$ define $E_i = \{x : x \in aw_i + b(B^n \cap W) \pmod 1\}$. Since $W = 0^k \times \mathbb{R}^r$, note that $E_i \subseteq 0^k \times [0, 1)^r$. By Claim 2 the set I is dense in $0^k \times [0, 1)^r$. Furthermore each set E_i has non-empty interior with respect to the subspace topology on $0^k \times [0, 1)^r$. Hence for all i, $1 \leq i \leq r+1$, we can find $n_i \in \mathbb{N}$ such that $n_i v' \pmod 1 \in E_i$.

Now $n_i v' \pmod 1 \in E_i$, implies that for some $\delta'_i \in E_i$, $n_i v' - \delta'_i \in \mathbb{Z}^n$. Furthermore $\delta'_i \in E_i$ implies that there exists $\delta_i \in aw_i + b(B^n \cap W)$ such that $\delta'_i - \delta_i \in \mathbb{Z}^n$. Hence $(n_i v' - \delta'_i) + (\delta'_i - \delta_i) = n_i v' - \delta_i \in \mathbb{Z}^n$. Let $a_i = n_i v' - \delta_i$. Note that $\|a_i - n_i v'\| = \|-\delta_i\| \leq a + b \leq 2a \leq 1/(4R)$. We claim that $\lfloor h_{K'}(a_i)\rfloor \leq h_{K'}(n_i v')$. First note that $h_{K'}(n_i v') = n_i b'$. Since we assume that $S \neq \emptyset$, we must have that $b' \in \mathbb{Z}$ and hence $n_i b' \in \mathbb{Z}$. Now note that

$$\begin{aligned} h_{K'}(a_i) &= h_{K'}((a_i - n_i v') + n_i v') \leq h_{K'}(n_i v') + h_{K'}(a_i - n_i v') \\ &= n_i b' + h_{K'}(-\delta_i) \\ &\leq n_i b' + h_{RB^n}(-\delta_i) \leq n_i b' + R\|\delta_i\| \leq n_i b' + R\left(\frac{1}{4R}\right) = n_i b' + \frac{1}{4}. \end{aligned}$$

Therefore we have that $\lfloor h_{K'}(a_i)\rfloor \leq \lfloor n_i b' + \frac{1}{4}\rfloor = n_i b' = h_{K'}(n_i v')$, since $n_i b' \in \mathbb{Z}$.

We claim that $\frac{a\epsilon}{4} B^n \cap W \subseteq \operatorname{conv}\{\delta_1, \ldots, \delta_{r+1}\}$. First note that by construction, $\operatorname{conv}\{\delta_1, \ldots, \delta_{r+1}\} \subseteq W$. Hence if the conclusion is false, then by the

separator theorem there exists $d \in W \cap S^{n-1}$ such that $h_{\frac{a\epsilon}{4}B^n \cap W}(d) = \frac{a\epsilon}{4} > \sup_{1\le i\le r+1}\langle d, \delta_i\rangle$. For each i, $1 \le i \le r+1$, we write $\delta_i = aw_i + bz_i$ where $\|z_i\| \le 1$. Now note that

$$\sup_{1\le i\le r+1} \langle d, \delta_i\rangle = \sup_{1\le i\le r+1} \langle d, aw_i + bz_i\rangle = \sup_{1\le i\le r+1} a\langle d, w_i\rangle + b\langle d, z_i\rangle$$
$$\ge \sup_{1\le i\le r+1} a\langle d, w_i\rangle - b\|d\|\|z_i\| \ge a\epsilon - b = \frac{a\epsilon}{2} > \frac{a\epsilon}{4},$$

a contradiction. Hence there exists $\lambda_1, \ldots, \lambda_{r+1} \ge 0$ and $\sum_{i=1}^{r+1} \lambda_i n_i = 1$ such that $\sum_{i=1}^{r+1} \lambda_i \delta_i = 0$.

Now we see that

$$\sum_{i=1}^{r+1} \lambda_i a_i = \sum_{i=1}^{r+1} \lambda_i n_i v' + \sum_{i=1}^{r+1} \lambda_i (a_i - n_i v') = \left(\sum_{i=1}^{r+1} \lambda_i n_i\right) v' - \sum_{i=1}^{r+1} \lambda_i \delta_i = \left(\sum_{i=1}^{r+1} \lambda_i n_i\right) v'. \tag{6}$$

Next note that

$$\sum_{i=1}^{r+1} \lambda_i \lfloor h_{K'}(a_i)\rfloor \le \sum_{i=1}^{r+1} \lambda_i h_{K'}(n_i v') = h_{K'}\left(\left(\sum_{i=1}^{r+1} \lambda_i n_i\right) v'\right). \tag{7}$$

Claim 4: Let $C = \{a_i\}_{i=1}^{r+1}$ for the a_i's from Claim 3. Then $CC(K, C) \cap \{x : \langle v', x\rangle = b'\} \subseteq \text{aff}(S)$.

Examine the set $P = \{x : \langle v', x\rangle = b', \langle a_i, x\rangle \le \lfloor h_{K'}(a_i)\rfloor, 1 \le i \le l+1\}$. From the proof of Claim 3, we know that for each i, $1 \le i \le r+1$, we have $\lfloor h_{K'}(a_i)\rfloor \le h_{K'}(n_i v') = n_i b'$ and hence $\langle n_i v' - a_i, x\rangle = \langle \delta_i, x\rangle \ge 0$, is a valid inequality for P. Now, from the proof of Claim 3, we have

$$\frac{a\epsilon}{4} B^n \cap W \subseteq \text{conv}\{\delta_1, \ldots, \delta_{r+1}\}. \tag{8}$$

We claim that for all $H \subseteq \{1, \ldots, r+1\}$, $|H| = r$, the set $\{\delta_i : i \in H\}$ is linearly independent. Assume not, then WLOG we may assume that $\delta_1, \ldots, \delta_r$ are not linearly independent. Hence there exists $d \in S^{n-1} \cap W$, such that $\langle d, \delta_i\rangle = 0$ for all $1 \le i \le n$. Now by possibly switching d to $-d$, we may assume that $\langle d, \delta_{r+1}\rangle \le 0$. Hence we get that $\sup_{1\le i\le r+1}\langle d, \delta_i\rangle \le 0$ in contradiction to (8).

Now let $\lambda_1, \ldots, \lambda_{r+1} \ge 0$, $\sum_{i=1}^{r+1} \lambda_i n_i = 1$ be a combination such that $\sum_{i=1}^{r+1} \lambda_i \delta_i = 0$. Note that $\lambda_1, \ldots, \lambda_{r+1}$ forms a linear dependency on $\delta_1, \ldots, \delta_{r+1}$, and hence by the previous claim we must have that $\lambda_i > 0$ for all $1 \le i \le r+1$.

We claim for $P \subseteq W^\perp$. To see this, note that $0 = \langle x, 0\rangle = \langle x, \sum_{i=1}^{r+1} \lambda_i \delta_i\rangle = \sum_{i=1}^{r+1} \lambda_i \langle x, \delta_i\rangle$ for every $x \in P$. Now since $\text{span}(\delta_1, \ldots, \delta_{r+1}) = W$, we see that $\langle x, \delta_i\rangle = 0$ for all $1 \le i \le r+1$ iff $x \in W^\perp$. Hence if $x \notin W^\perp$, then by the above equation and the fact that $\lambda_i > 0$ for all $i \in \{1, \ldots, r+1\}$, there exists $i, j \in \{1, \ldots, r+1\}$ such that $\langle x, \delta_i\rangle > 0$ and $\langle x, \delta_j\rangle < 0$. But then $x \notin P$, since $\langle x, \delta_j\rangle < 0$, a contradiction. Now $W = 0^k \times \mathbb{R}^r$, hence $W^\perp = \mathbb{R}^k \times 0^r$. To complete the proof we see that $P \subseteq \{x : x \in \mathbb{R}^k \times 0^r, \langle v', x\rangle = b'\} = \text{aff}(S)$.

3.3 Lifting the CG Closure of an Exposed Face of K

Proposition 3. *Let $K \subseteq \mathbb{R}^n$ be a compact convex set. Take $v \in \mathbb{R}^n$. Assume that $CC(F_v(K))$ is finitely generated. Then $\exists\ S \subseteq \mathbb{Z}^n$, $|S| < \infty$, such that $CC(K,S)$ is a polytope and*

$$CC(K,S) \cap H_v^=(K) = CC(F_v(K)) \tag{9}$$
$$CC(K,S) \subseteq H_v. \tag{10}$$

Proof. The right to left containment in (9) is direct from $CC(F_v(K)) \subseteq CC(K,S)$ as every CG cut for K is a CG cut for $F_v(K)$. For the reverse containment and for (10) we proceed as follows.

Using Proposition 2 there exists $S_1 \subseteq \mathbb{Z}^n$ such that $CC(K,S_1) \cap H_v^=(K) \subseteq \text{aff}_I(H_v^=(K))$ and $CC(K,S_1) \subseteq \{x \in \mathbb{R}^n : \langle v,x\rangle \leq h_K(v)\}$. Next let $G \subseteq \mathbb{Z}^n$ be such that $CC(F_v(K),G) = CC(F_v(K))$. For each $w \in G$, by Proposition 1 there exists $w' \in \mathbb{Z}^n$ such that $CC(K,w') \cap \text{aff}_I(H_v^=(K)) \subseteq CC(F_v(K),w) \cap \text{aff}_I(H_v^=(K))$. For each $w \in G$, add w' above to S_2. Now note that

$$\begin{aligned} CC(K,S_1 \cup S_2) \cap H_v^=(K) &= CC(K,S_1) \cap CC(K,S_2) \cap H_v^=(K) \\ &\subseteq CC(K,S_2) \cap \text{aff}_I(H_v^=(K)) \\ &= CC(F_v(K),G) \cap \text{aff}(H_v^=(K)) \subseteq CC(F_v(K)). \end{aligned}$$

Now let $S_3 = \{\pm e_i : 1 \leq i \leq n\}$. Note that since K is compact $CC(K,S_3)$ is a cuboid with bounded side lengths, and hence is a polytope. Letting $S = S_1 \cup S_2 \cup S_3$, yields the desired result.

We now obtain a generalization of the classical result known for rational polyhedra.

Corollary 1. *Let K be a compact convex set and let F be an exposed face of K, then we have that $CC(F) = CC(K) \cap F$.*

4 Approximation of the CG Closure

4.1 Approximation 1 of the CG Closure

In this section, we construct a first approximation of the CG closure of K. Under the assumption that the CG closure of every proper exposed face is finitely generated, we use a compactness argument to construct a finite set of CG cuts $S \subseteq \mathbb{Z}^n$ such that $CC(K,S) \subseteq K \cap \text{aff}_I(K)$. We use the following lemma (stated without proof) to simplify the analysis of integral affine subspaces.

Lemma 4. *Take $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Then there exists $\lambda \in \mathbb{R}^m$ such that for $a' = \lambda A$, $b' = \lambda b$, we have that $\{x \in \mathbb{Z}^n : Ax = b\} = \{x \in \mathbb{Z}^n : a'x = b'\}$.*

Proposition 4. *Let $\emptyset \neq K \subseteq \mathbb{R}^n$ be a compact convex set. If $CC(F_v(K))$ is finitely generated for any proper exposed face $F_v(K)$ then $\exists\ S \subseteq \mathbb{Z}^n$, $|S| < \infty$, such that $CC(K,S) \subseteq K \cap \text{aff}_I(K)$ and $CC(K,S)$ is a polytope.*

Proof. Let us express $\text{aff}(K)$ as $\{x \in \mathbb{R}^n : Ax = b\}$. Note that $\text{aff}(K) \neq \emptyset$ since $K \neq \emptyset$. By Lemma 4 there exists λ, $c = \lambda A$ and $d = \lambda b$, and such that $\text{aff}(K) \cap \mathbb{Z}^n = \{x \in \mathbb{Z}^n : \langle c, x\rangle = b\}$. Since $h_K(c) = b$ and $h_K(-c) = -b$, using Proposition 2 on c and $-c$, we can find $S_A \subseteq \mathbb{Z}^n$ such that $CC(K, S_A) \subseteq \text{aff}(\{x \in \mathbb{Z}^n : \langle c, x\rangle = b\}) = \text{aff}_I(K)$.

Express $\text{aff}(K)$ as $W + a$, where $W \subseteq \mathbb{R}^n$ is a linear subspace and $a \in \mathbb{R}^n$. Now take $v \in W \cap S^{n-1}$. Note that $F_v(K)$ is a proper exposed face and hence, by assumption, $CC(F_v(K))$ is finitely generated. Hence by Proposition 3 there exists $S_v \subseteq \mathbb{Z}^n$ such that $CC(K, S_v)$ is a polytope, $CC(K, S_v) \cap H_v^=(K) = CC(F_v(K))$ and $CC(K, S_v) \subseteq \{x \in \mathbb{R}^n : \langle x, v\rangle \leq h_K(v)\}$. Let $K_v = CC(K, S_v)$, then we have the following claim whose proof we skip because of lack of space.

Claim: $\exists$ open neighborhood N_v of v in $W \cap S^{n-1}$ such that $v' \in N_v \Rightarrow h_{K_v}(v') \leq h_K(v')$.

Note that $\{N_v : v \in W \cap S^{n-1}\}$ forms an open cover of $W \cap S^{n-1}$, and since $W \cap S^{n-1}$ is compact, there exists a finite subcover $N_{v_1}, \ldots, N_{v_k}$ such that $\bigcup_{i=1}^k N_{v_i} = W \cap S^{n-1}$. Now let $S = S_A \cup \cup_{i=1}^k S_{v_i}$. We claim that $CC(K, S) \subseteq K$. Assume not, then there exists $x \in CC(K, S) \backslash K$. Since $CC(K, S) \subseteq CC(K, S_A) \subseteq W + a$ and $K \subseteq W + a$, by the separator theorem there exists $w \in W \cap S^{n-1}$ such that $h_K(w) = \sup_{y \in K} \langle y, w\rangle < \langle x, w\rangle \leq h_{CC(K,S)}(w)$. Since $w \in W \cap S^{n-1}$, there exists i, $1 \leq i \leq k$, such that $w \in N_{v_i}$. Note then we obtain that $h_{CC(K,S)}(w) \leq h_{CC(K,S_{v_i})}(w) = h_{K_{v_i}}(w) \leq h_K(w)$, a contradiction. Hence $CC(K, S) \subseteq K$ as claimed. $CC(K, S)$ is a polytope because it is the intersection of polyhedra of which at least one is a polytope.

4.2 Approximation 2 of the CG Closure

In this section, we augment the first approximation of the $CC(K)$ with a finite number of extra CG cuts so that this second approximation matches $CC(K)$ on the relative boundary of K.

To achieve this, we observe that our first approximation of $CC(K)$ is polyhedral and contained in K, and hence its intersection with the relative boundary of K is contained in the union of a finite number of proper exposed faces of K. Therefore, by applying Proposition 3 to each such face (i.e. adding their lifted CG closure), we can match $CC(K)$ on the relative boundary as required. The following lemma (stated without proof) makes precise the previous statements.

Lemma 5. *Let $K \subseteq \mathbb{R}^n$ be a convex set and $P \subseteq K$ be a polytope. Then there exists $F_{v_1}, \ldots, F_{v_k} \subseteq K$, proper exposed faces of K, such that $P \cap \text{relbd}(K) \subseteq \bigcup_{i=1}^k F_{v_i}$*

Proposition 5. *Let $K \subseteq \mathbb{R}^n$ be a compact convex set. If $CC(F_v)$ is finitely generated for any proper exposed face F_v then $\exists$ $S \subseteq \mathbb{Z}^n$, $|S| < \infty$, such that*

$$CC(K, S) \subseteq K \cap \text{aff}_I(K) \tag{11}$$

$$CC(K, S) \cap \text{relbd}(K) = CC(K) \cap \text{relbd}(K) \tag{12}$$

Proof. By Proposition 4, there exists $S_I \subseteq \mathbb{Z}^n$, $|S_I| < \infty$, such that $CC(K, S_I) \subseteq K \cap \text{aff}_I(K)$ and $CC(K, S_I)$ is a polytope. Since $CC(K, S_I) \subseteq K$ is a polytope, let $F_{v_1}, \ldots, F_{v_k}$ be the proper exposed faces of K given by Lemma 5. By Proposition 3, there exists $S_i \subseteq \mathbb{Z}^n$, $|S_i| < \infty$, such that $CC(K, S_i) \cap H_{v_i} = CC(F_{v_i})$. Let $S = S_I \cup \cup_{i=1}^k S_i$. We claim that $CC(K, S) \cap \text{relbd}(K) \subseteq CC(K) \cap \text{relbd}(K)$. For this note that $x \in CC(K, S) \cap \text{relbd}(K)$ implies $x \in CC(K, S_I) \cap \text{relbd}(K)$, and hence there exists i, $1 \leq i \leq k$, such that $x \in F_{v_i}$. Then $x \in CC(K, S) \cap H_{v_i} \subseteq CC(K, S_i) \cap H_{v_i} = CC(F_{v_i}) \subseteq CC(K) \cap \text{relbd}(K)$. The reverse inclusion is direct.

5 Proof of Theorem

Finally, we have all the ingredients to prove the main result of this paper. The proof is by induction on the dimension of K. Trivially, the result holds for zero dimensional convex bodies. Now using the induction hypothesis, we can construct the second approximation of $CC(K)$ using Proposition 5 (since it assumes that the CG closure of every exposed face is finitely generated). Lastly, we observe that any CG cut for K not dominated by those already considered in the second approximation of $CC(K)$ must separate a vertex of this approximation lying in the relative interior of K. From here, it is not difficult to show that only a finite number of such cuts exists, thereby proving the polyhedrality of $CC(K)$. The proof here is similar to the one used for strictly convex sets, with the additional technicality that here $\text{aff}(K)$ may be irrational.

Theorem 4. *Let $K \subseteq \mathbb{R}^n$ be a non-empty compact convex set. Then $CC(K)$ is finitely generated.*

Proof. We proceed by induction on the affine dimension of K. For the base case, $\dim(\text{aff}(K)) = 0$, i.e. $K = \{x\}$ is a single point. Here it is easy to see that setting $S = \{\pm e_i : i \in \{1, \ldots, n\}\}$, we get that $CC(K, S) = CC(K)$. The base case thus holds.

Now for the inductive step let $0 \leq k < n$ let K be a compact convex set where $\dim(\text{aff}(K)) = k + 1$ and assume the result holds for sets of lower dimension. By the induction hypothesis, we know that $CC(F_v)$ is finitely generated for every proper exposed face F_v of K, since $\dim(F_v) \leq k$. By Proposition 5, there exists a set $S \subseteq \mathbb{Z}^n$, $|S| < \infty$, such that (11) and (12) hold. If $CC(K, S) = \emptyset$, then we are done. So assume that $CC(K, S) \neq \emptyset$. Let $A = \text{aff}_I(K)$. Since $CC(K, S) \neq \emptyset$, we have that $A \neq \emptyset$ (by (11)), and so we may pick $t \in A \cap \mathbb{Z}^n$. Note that $A - t = W$, where W is a linear subspace of $\mathbb{R}^n$ satisfying $W = \text{span}(W \cap \mathbb{Z}^n)$. Let $L = W \cap \mathbb{Z}^n$. Since $t \in \mathbb{Z}^n$, we easily see that $CC(K - t, T) = CC(K, T) - t$ for all $T \subseteq \mathbb{Z}^n$. Therefore $CC(K)$ is finitely generated iff $CC(K - t)$ is. Hence replacing K by $K - t$, we may assume that $\text{aff}_I(K) = W$.

Let π_W denote the orthogonal projection onto W. Note that for all $x \in W$, and $z \in \mathbb{Z}^n$, we have that $\langle z, x \rangle = \langle \pi_W(z), x \rangle$. Now since $CC(K, S) \subseteq K \cap W$, we see that for all $z \in \mathbb{Z}^n$, $CC(K, S \cup \{z\}) = CC(K, S) \cap \{x : \langle z, x \rangle \leq \lfloor h_K(z) \rfloor\} = CC(K, S) \cap \{x : \langle \pi_W(z), x \rangle \leq \lfloor h_K(z) \rfloor\}$. Let $L^* = \pi_W(\mathbb{Z}^n)$. Since W is a rational

subspace, we have that L^* is full dimensional lattice in W. Now fix an element of $w \in L^*$ and examine $V_w := \{\lfloor h_K(z) \rfloor : \pi_W(z) = w, z \in \mathbb{Z}^n\}$. Note that $V_w \subseteq \mathbb{Z}$. We claim that $\inf(V_w) \geq -\infty$. To see this, note that

$$\begin{aligned}\inf\{\lfloor h_K(z) \rfloor : \pi_W(z) = w, z \in \mathbb{Z}^n\} &\geq \inf\{\lfloor h_{K\cap W}(z) \rfloor : \pi_W(z) = w, z \in \mathbb{Z}^n\} \\ &= \inf\{\lfloor h_{K\cap W}(\pi_W(z)) \rfloor : \pi_W(z) = w, z \in \mathbb{Z}^n\} \\ &= \lfloor h_{K\cap W}(w) \rfloor > -\infty.\end{aligned}$$

Now since V_w is a lower bounded set of integers, there exists $z_w \in \pi_W^{-1}(w) \cap \mathbb{Z}^n$ such that $\inf(V_w) = \lfloor h_K(z_w) \rfloor$. From the above reasoning, we see that $CC(K, S \cup \pi_W^{-1}(z) \cap \mathbb{Z}^n) = CC(K, S \cup \{z_w\})$. Now examine the set $C = \{w : w \in L^*, CC(K, S \cup \{z_w\}) \subsetneq CC(K, S)\}$. Here we get that

$$CC(K) = CC(K, S \cup \mathbb{Z}^n) = CC(K, S \cup \{z_w : w \in L^*\}) = CC(K, S \cup \{z_w : w \in C\}).$$

From the above equation, if we show that $|C| < \infty$, then $CC(K)$ is finitely generated. To do this, we will show that there exists $R > 0$, such that $C \subseteq RB^n$, and hence $C \subseteq L^* \cap RB^n$. Since L^* is a lattice, $|L^* \cap RB^n| < \infty$ for any fixed R, and so we are done.

Now let $P = CC(K, S)$. Since P is a polytope, we have that $P = \text{conv}(\text{ext}(P))$. Let $I = \text{ext}(P) \cap \text{relint}(K)$, and let $B = \text{ext}(P) \cap \text{relbd}(K)$. Hence $\text{ext}(P) = I \cup B$. By assumption on $CC(K, S)$, we know that for all $v \in B$, we have that $v \in CC(K)$. Hence for all $z \in \mathbb{Z}^n$, we must have that $\langle z, v \rangle \leq \lfloor h_K(z) \rfloor$ for all $v \in B$. Now assume that for some $z \in \mathbb{Z}^n$, $CC(K, S \cup \{z\}) \subsetneq CC(K, S) = P$. We claim that $\langle z, v \rangle > \lfloor h_K(z) \rfloor$ for some $v \in I$. If not, then $\langle v, z \rangle \leq \lfloor h_K(z) \rfloor$ for all $v \in \text{ext}(P)$, and hence $CC(K, S \cup \{z\}) = CC(K, S)$, a contradiction. Hence such a $v \in I$ must exist.

For $z \in \mathbb{Z}^n$, note that $h_K(z) \geq h_{K\cap W}(z) = h_{K\cap W}(\pi_W(z))$. Hence $\langle z, v \rangle > \lfloor h_K(z) \rfloor$ for $v \in I$ only if $\langle \pi_W(z), v \rangle = \langle z, v \rangle > \lfloor h_{K\cap W}(\pi_W(z)) \rfloor$. Let $C' := \{w \in L^* : \exists v \in I, \langle v, w \rangle > \lfloor h_{K\cap W} \rfloor(w)\}$. From the previous discussion, we see that $C \subseteq C'$.

Since $I \subseteq \text{relint}(K) \cap W = \text{relint}(K \cap W)$ we have $\delta_v = \sup\{r \geq 0 : rB^n \cap W + v \subseteq K \cap W\} > 0$ for all $v \in I$. Let $\delta = \inf_{v\in I} \delta_v$. Since $|I| < \infty$, we see that $\delta > 0$. Now let $R = \frac{1}{\delta}$. Take $w \in L^*$, $\|w\| \geq R$. Note that $\forall v \in I$,

$$\lfloor h_{K\cap W}(w) \rfloor \geq h_{K\cap W}(w) - 1 \geq h_{(v+\delta B^n)\cap W}(w) - 1 = \langle v, w \rangle + \delta\|w\| - 1 \geq \langle v, w \rangle.$$

Hence $w \notin C'$. Therefore $C \subseteq C' \subseteq RB^n$ and $CC(K)$ is finitely generated.

6 Remarks

Using techniques developed in Proposition 2 and Lemma 4 it is possible to prove the following.

Theorem 5. *Let $T = \{x \in \mathbb{R}^n : Ax = b\}$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. The following holds:*

1. *If* $\text{aff}_I(T) = \emptyset$*, then for all* $D > 0$ *there exists* $z \in \mathbb{Z}^n$ *such that* $CC(T \cap DB^n, \{z, -z\}) = \emptyset$.
2. *If* $\text{aff}_I(T) \neq \emptyset$*, then for all* $D > 0$ *there exists* $S \subseteq \mathbb{Z}^n$*,* $|S| = n - dim(\text{aff}_I(T)) + 1$ *such that* $CC(T \cap DB^n, S) = \text{aff}_I(T)$.

The above result can be considered as a generalization of Integer Farkas' Lemma: If A and b are rational and $\text{aff}_I(T) = \emptyset$, then it can be shown (we skip details due to lack of space) if $D > 0$ is sufficiently large, then $CC(T \cap DB^n, \{z, -z\}) = \emptyset$ implies that $CC(T, \{z, -z\}) = \emptyset$ which is one half of regular Integer Farkas' Lemma.

References

1. Ben-Tal, A., Nemirovski, A.: Lectures on modern convex optimization: analysis, algorithms, and engineering applications. Society for Industrial and Applied Mathematics, Philadelphia, PA (2001)
2. Bonami, P., Dash, G.C.S., Fischetti, M., Lodi, A.: Projected Chvatal-Gomory Cuts for Mixed Integer Linear Programs. Mathematical Programming 113, 241–257 (2008)
3. Cassels, J.W.S.: An introduction to Diophantine approximation. Hafner, New York (1972)
4. Çezik, M.T., Iyengar, G.: Cuts for mixed 0-1 conic programming. Mathematical Programming 104, 179–202 (2005)
5. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Mathematics 4, 305–337 (1973)
6. Dadush, D., Dey, S.S., Vielma, J.P.: The Chvátal-Gomory Closure of Strictly Convex Body (2010) (to appear in Mathematics of Operations Research)
7. Dey, S.S., Vielma, J.P.: The Chvátal-Gomory Closure of an Ellipsoid Is a Polyhedron. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 327–340. Springer, Heidelberg (2010)
8. Dunkel, J., Schulz, A.S.: The Gomory-chvátal closure of a non-rational polytope is a rational polytope (2010), `http://www.optimization-online.org/DB_HTML/2010/11/2803.html`
9. Edmonds, J.: Paths, trees, and flowers. Canadian Journal of mathematics 17, 449–467 (1965)
10. Fischetti, M., Lodi, A.: Optimizing over the first Chvátal closure. Mathematical Programming, Series B 110, 3–20 (2007)
11. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bulletin of the American Mathematical Society 64, 275–278 (1958)
12. Grötschel, M., Padberg, M.: On the symmetric travelling salesman problem I: Inequalities. Math. Programming 16, 265–280 (1979)
13. Grötschel, M., Padberg, M.: On the symmetric travelling salesman problem II: Lifting theorems and facets. Math. Programming 16, 281–302 (1979)
14. Niven, I.M.: Diophantine approximations. Interscience Publishers, New York (1963)
15. Schrijver, A.: On cutting planes. Annals of Discrete Mathematics 9, 291–296 (1980); combinatorics 79 (Proc. Colloq., Univ. Montréal, Montreal, Que., 1979), Part II

Design and Verify: A New Scheme for Generating Cutting-Planes

Santanu S. Dey[1] and Sebastian Pokutta[2]

[1] H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
santanu.dey@isye.gatech.edu
[2] Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
pokutta@mit.edu

Abstract. A cutting-plane procedure for integer programming (IP) problems usually involves invoking a black-box procedure (such as the Gomory-Chvátal (GC) procedure) to *compute* a cutting-plane. In this paper, we describe an alternative paradigm of using the same cutting-plane black-box. This involves two steps. In the first step, we *design* an inequality $cx \leq d$, *independent* of the cutting-plane black-box. In the second step, we *verify* that the designed inequality is a valid inequality by verifying that the set $P \cap \{x \in \mathbb{R}^n : cx \geq d+1\} \cap \mathbb{Z}^n$ is empty using cutting-planes from the black-box. Here P is the feasible region of the linear-programming relaxation of the IP. We refer to the closure of all cutting-planes that can be verified to be valid using a specific cutting-plane black-box as the *verification closure* of the considered cutting-plane black-box. This paper conducts a systematic study of properties of verification closures of various cutting-plane black-box procedures.

Keywords: Verification Scheme, Cutting-Planes, Integer Programming.

1 Introduction

Cutting-planes are a crucial tool in solving Integer Programs (IPs). Often the only guiding principal (example: Kelley's Method [13]) used in deriving generic cutting-planes (like Gomory-Chvátal or split cuts) is that the incumbent fractional point must be separated. Therefore, cutting-planes are generated 'almost blindly', where we apply some black-box method to *constructively compute* valid cutting-planes and hope for the right set of cuts to appear that helps in proving optimality (or close significant portion of the gap). Now if we were somehow able to *deliberately design strong cutting-planes* that were tailor-made, for example, to prove the optimality of good candidate solutions, then we could possibly speed up IP solvers. This motivates a different paradigm to generate valid cutting-planes for integer programs: First we *design* a useful cutting-plane without considering its validity. Then, once the cutting-plane is designed, we *verify* that it is valid.

For $n \in \mathbb{N}$, let $[n] = \{1, ..., n\}$ and for a polytope $P \subseteq \mathbb{R}^n$ denote its *integral hull* by $P_I := \text{conv}\,(P \cap \mathbb{Z}^n)$. We now precisely describe these verification

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 143–155, 2011.

schemes (abbreviated as: $\mathbb{V}$-schemes). Let M be an *admissible* cutting-plane procedure (i.e., a valid and 'reasonable' cutting-plane system - we will formally define these) and let $\mathrm{M}(P)$ be the closure with respect to the family of cutting-planes obtained using M. For example, M could represent split cuts and then $\mathrm{M}(P)$ represents the split closure of P. Usually using cutting-planes from a cutting-plane procedure M, implies using valid inequalities for $\mathrm{M}(P)$ as cutting-planes. In the $\mathbb{V}$-scheme, we apply the following procedure: We design or guess the inequality $cx \leq d$ where $(c, d) \in \mathbb{Z}^n \times \mathbb{Z}$. To verify that this inequality is valid for P_I, we apply M to $P \cap \{x \in \mathbb{R}^n \mid cx \geq d+1\}$ and check whether $\mathrm{M}(P \cap \{x \in \mathbb{R}^n \mid cx \geq d+1\}) = \emptyset$. If $\mathrm{M}(P \cap \{x \in \mathbb{R}^n \mid cx \geq d+1\}) = \emptyset$, then $cx \leq d$ is a valid inequality for P_I and we say it can be obtained using the $\mathbb{V}$-scheme of M.

We might wonder how much we gain from having to only *verify* that a given inequality $cx \leq d$ is valid for P_I, rather than actually *computing* it. In fact at a first glance, it is not even clear that there would be any difference between computing and verifying. The strength of the verification scheme lies in the following inclusion that can be readily verified for admissible cutting-plane procedures:

$$\mathrm{M}(P \cap \{x \in \mathbb{R}^n \mid cx \geq d+1\}) \subseteq \mathrm{M}(P) \cap \{x \in \mathbb{R}^n \mid cx \geq d+1\}\,. \tag{1}$$

The interpretation of this inclusion is that an additional inequality $cx \geq d+1$ appended to the description of P can provide us with crucial extra information when deriving new cutting-planes that is not available when considering P alone and then adding the additional inequality afterwards to the M-closure of P. In other words, (1) can potentially be a strict inclusion such that $\mathrm{M}(P \cap \{x \in \mathbb{R}^n \mid cx \geq d+1\}) = \emptyset$ while $\mathrm{M}(P) \cap \{x \in \mathbb{R}^n \mid cx \geq d+1\} \neq \emptyset$. This is equivalent to saying that we can *verify* the validity of $cx \leq d$, however we are not able to *compute* $cx \leq d$. To the best of our knowledge, the only paper discussing a related idea is [4], but theoretical and computational potential of this approach has not been further investigated.

The set obtained by intersecting all cutting-planes that can be verified to be valid using M will be called the verification closure (abbreviated as: $\mathbb{V}$-closure) of M and denoted by $\partial\mathrm{M}(P)$. Formally,

$$\partial\mathrm{M}(P) := \bigcap_{\substack{(c,d)\in\mathbb{Z}^n\times\mathbb{Z} \\ \text{s.t. } \mathrm{M}(P\cap\{x\in\mathbb{R}^n \mid cx\geq d+1\})=\emptyset}} \{x \in \mathbb{R}^n \mid cx \leq d\}\,. \tag{2}$$

Under mild conditions (1) implies $\partial\mathrm{M}(P) \subseteq \mathrm{M}(P)$ for all rational polytopes P. Since there exist inequalities that can be verified but not computed, this inclusion can be proper.

Outline and contribution. This paper conducts a systematic study of the strengths and weaknesses of the $\mathbb{V}$-schemes. In Section 2, we prove basic properties of the $\mathbb{V}$-closure. In order to present these results, we first describe general classes of reasonable cutting-planes, the so called *admissible cutting-plane procedures*, a machinery developed in [17]. We prove that ∂M is *almost admissible*, i.e.

the $\mathbb{V}$-schemes satisfy many important properties that all known classes of admissible cutting-plane procedures including Gomory-Chvátal (GC) cuts [12,2], lift-and-project cuts [1], split cuts (SC) [6], and $\mathrm{N}, \mathrm{N}_0, \mathrm{N}_+$ [14] cuts satisfy.

In Section 3, we show first that $\mathbb{V}$-schemes have natural inherent strength, i.e., even if M is an arbitrarily weak admissible cutting-plane procedure, $\partial\mathrm{M}(P)$ is at least as strong as the GC and the N_0 closures. We then compare the strength of various regular closures (GC cuts, split cuts, and $\mathrm{N}_0, \mathrm{N}, \mathrm{N}_+$ cuts) with their $\mathbb{V}$-versions and with each other. For example, we show that $\partial\mathrm{GC}(P) \subseteq \mathrm{SC}(P)$ and $\partial\mathrm{N}_0(P) \subseteq \mathrm{SC}(P)$. The complete list of these results is illustrated in Figure 1.

In Section 4, we present upper and lower bounds on the rank of valid inequalities with respect to the $\mathbb{V}$-closures for a large class of 0/1 problems. These results show that while the $\mathbb{V}$-closures are strong, they not unrealistically so.

In Section 5, we illustrate the strength of the $\mathbb{V}$-schemes on specific structured problems. We show that facet-defining inequalities of monotone polytopes contained in $[0,1]^n$ have low rank with respect to any $\partial\mathrm{M}$ operator. We show that numerous families of inequalities with high GC, N_0, or N rank [14] (such as clique inequalities) for the *stable set polytope* have a rank of 1 with respect to any $\partial\mathrm{M}$ with M being arbitrarily weak and admissible. We will also show that for the subtour elimination relaxation of the *traveling salesman problem* the rank for $\partial\mathrm{M}$ with $\mathrm{M} \in \{\mathrm{GC}, \mathrm{SC}, N_0, N, N_+\}$ is in $\Theta(n)$ where n is the number of nodes, i.e., the rank is $\Theta(\sqrt{\dim(P)})$ with P being the TSP-polytope. It is well-known that for the case of *general polytopes in* $\mathbb{R}^2$ the GC rank can be arbitrarily large. In contrast, we establish that the rank of general polytopes in $\mathbb{R}^2$ with respect to $\partial\mathrm{GC}$ is 1.

2 General Properties of the $\mathbb{V}$-Closure

For the ease of presentation, we will only consider *rational* polytopes in the following definition, although it readily generalizes to compact convex sets.

Definition 1 ([17]). *A cutting-plane procedure* M *defined for polytopes* $P := \{x \in [0,1]^n \mid Ax \leq b\}$ *is* admissible *if the following holds:*

1. Validity*:* $P_I \subseteq \mathrm{M}(P) \subseteq P$.
2. Inclusion preservation*: If* $P \subseteq Q$*, then* $\mathrm{M}(P) \subseteq \mathrm{M}(Q)$ *for all polytopes* $P, Q \subseteq [0,1]^n$.
3. Homogeneity*:* $\mathrm{M}(F \cap P) = F \cap \mathrm{M}(P)$*, for all faces* F *of* $[0,1]^n$.
4. Single coordinate rounding*: If* $x_i \leq \epsilon < 1$ *(or* $x_i \geq \epsilon > 0$*) is valid for* P*, then* $x_i \leq 0$ *(or* $x_i \geq 1$*) is valid for* $\mathrm{M}(P)$.
5. Commuting with coordinate flips and duplications*:* $\tau_i(\mathrm{M}(P)) = \mathrm{M}(\tau_i(P))$*, where* τ_i *is either one of the following two operations: (i) Coordinate flip:* $\tau_i : [0,1]^n \to [0,1]^n$ *with* $(\tau_i(x))_i = (1 - x_i)$ *and* $(\tau_i(x))_j = x_j$ *for* $j \in [n] \setminus \{i\}$*; (ii) Coordinate Duplication:* $\tau_i : [0,1]^n \to [0,1]^{n+1}$ *with* $(\tau_i(x))_{n+1} = x_i$ *and* $(\tau_i(x))_j = x_j$ *for* $j \in [n]$.
6. Substitution independence*: Let* φ_F *be the projection onto the face* F *of* $[0,1]^n$*. Then* $\varphi_F(\mathrm{M}(P \cap F)) = \mathrm{M}(\varphi_F(P \cap F))$.

7. Short verification: *There exists a polynomial p such that for any inequality $cx \leq d$ that is valid for $\mathrm{M}(P)$ there is a set $I \subseteq [m]$ with $|I| \leq p(n)$ such that $cx \leq d$ is valid for $\mathrm{M}(\{x \in \mathbb{R}^n \mid a_i x \leq b_i, i \in I\})$. We call $p(n)$ the* verification degree *of* M.

If M *is defined for general rational polytopes $P \subseteq \mathbb{R}^n$, then we say* M *is admissible if (A.)* M *satisfies (1.)-(7.) when restricted to polytopes contained in $[0,1]^n$ and (B.) for general polytopes $P \subseteq \mathbb{R}^n$,* M *satisfies (1.), (2.), (7.) and Homogeneity is replaced by*

8. Strong Homogeneity: *If $P \subseteq F^{\leq} := \{x \in \mathbb{R}^n \mid ax \leq b\}$ and $F = \{x \in \mathbb{R}^n \mid ax = b\}$ where $(a,b) \in \mathbb{Z}^n \times \mathbb{Z}$, then $\mathrm{M}(F \cap P) = \mathrm{M}(P) \cap F$.*

In the following, we assume that $\mathrm{M}(P)$ is a closed convex set. If M *satisfies all required properties for being admissible except (7.), then we say* M *is* almost admissible.

Requiring strong homogeneity in the general case leads to a slightly more restricted class than the requirement of homogeneity in the 0/1 case. We note here that almost all known classes of cutting-plane schemes such as GC cuts, lift-and-project cuts, split cuts, and N, N_0, N_+ are admissible (cf. [17] for more details). Observe that (1) in Section 1 follows from inclusion preservation.

Next we present a technical lemma that we require for the main result of this section. We will use $\{\alpha x \leq \beta\}$ as a shorthand for $\{x \in \mathbb{R}^n \mid \alpha x \leq \beta\}$.

Lemma 1. *Let Q be a compact set contained in the interior of the set $\{\beta x \leq \zeta\}$ with $(\beta, \zeta) \in \mathbb{Z}^n \times \mathbb{Z}$ and let $(\alpha, \eta) \in \mathbb{Z}^n \times \mathbb{Z}$. Then there exists a positive integer τ such that Q is strictly contained in the set $\{(\alpha + \tau\beta)x \leq \eta + \tau\zeta\}$.*

We next show that ∂M satisfies almost all properties that we should expect from a well-defined cutting-plane procedure.

Theorem 1. *Let* M *be an admissible cutting-plane procedure. Then* ∂M *is almost admissible. In particular,*

1. *For 0/1 polytopes,* ∂M *satisfies properties (1.) to (6.).*
2. *If* M *is defined for general polytopes, then* ∂M *satisfies property (8.).*

Proof. It is straightforward to verify (1.), (2.), and (4.) - (6.). The non-trivial part is property (8.) (or (3.) respectively). In fact it follows from the original operator M having this property. We will prove (8.); property (3.) in the case of $P \subseteq [0,1]^n$ follows *mutatis mutandis*.

First observe that $\partial\mathrm{M}(P \cap F) \subseteq \partial\mathrm{M}(P)$ and $\partial\mathrm{M}(P \cap F) \subseteq F$. Therefore, $\partial\mathrm{M}(P \cap F) \subseteq \partial\mathrm{M}(P) \cap F$. To verify $\partial\mathrm{M}(P \cap F) \supseteq \partial\mathrm{M}(P) \cap F$, we show that if $\hat{x} \notin \partial\mathrm{M}(P \cap F)$, then $\hat{x} \notin \partial\mathrm{M}(P) \cap F$. Observe first that if $\hat{x} \notin P \cap F$, then $\hat{x} \notin \partial\mathrm{M}(P) \cap F$. Therefore, we assume that $\hat{x} \in P \cap F$. Hence we need to prove that if $\hat{x} \notin \partial\mathrm{M}(P \cap F)$ and $\hat{x} \in P \cap F$, then $\hat{x} \notin \partial\mathrm{M}(P)$. Since $\hat{x} \notin \partial\mathrm{M}(P \cap F)$, there exists $c \in \mathbb{Z}^n$ and $d \in \mathbb{Z}$ such that $c\hat{x} > d$ and $\mathrm{M}(P \cap F \cap \{cx \geq d+1\}) = \emptyset$. By strong homogeneity of M, we obtain

$$\mathrm{M}(P \cap \{cx \geq d+1\}) \cap F = \emptyset. \tag{3}$$

Let $F^{\leq} = \{ax \leq b\}$ and $F = \{ax = b\}$ with $P \subseteq F^{\leq}$. Now observe that (3) is equivalent to saying that $\mathrm{M}(P \cap \{cx \geq d+1\})$ is contained in the interior of the set $\{ax \leq b\}$. Therefore by Lemma 1, there exists a $\tau \in \mathbb{Z}_+$ such that $\mathrm{M}(P \cap \{cx \geq d+1\})$ is contained in the interior of $\{(c+\tau a)x \leq d+1+\tau b\}$. Equivalently, $\mathrm{M}(P \cap \{cx \geq d+1\}) \cap \{(c+\tau a)x \geq d+1+\tau b\} = \emptyset$ which implies

$$\mathrm{M}(P \cap \{cx \geq d+1\}) \cap (P \cap \{(c+\tau a)x \geq d+1+\tau b\}) = \emptyset. \tag{4}$$

Since $P \subseteq F^{\leq}$, we obtain that

$$P \cap \{(c+\tau a)x \geq d+1+\tau b\} \subseteq P \cap \{cx \geq d+1\}. \tag{5}$$

Now using (4), (5) and the inclusion preservation property of M it follows that $\mathrm{M}(P \cap \{(c+\tau a)x \geq d+1+\tau b\}) = \emptyset$. Thus $(c+\tau a)x \leq d+\tau b$ is a valid inequality for $\partial\mathrm{M}(P)$. Moreover note that since $\hat{x} \in P \cap F$, we have that $a\hat{x} = b$. Therefore, $(c+\tau a)\hat{x} = c\hat{x} + \tau b > d + \tau b$, where the last inequality follows from the fact that $c\hat{x} > d$. □

It can be shown that short verification, i.e., property (7.) of admissible systems follows whenever $\partial\mathrm{M}(P)$ is a rational polyhedron. However, we do not need this property for the results in this paper.

3 Strength and Comparisons of $\mathbb{V}$-Closures

In this section, we compare various regular closures and their verification counterparts with each other. We first formally define possible relations between admissible closures and the notation we use.

Definition 2. *Let* L, M *be almost admissible. Then*

1. L refines M, *if for all polytopes* P *we have* $\mathrm{L}(P) \subseteq \mathrm{M}(P)$. *We write:* $\mathrm{L} \subseteq \mathrm{M}$. *It is indicated by empty arrow heads in Figure 1.*
2. L strictly refines M, *if* L *refines* M *and there exists a polytope* P *such that* $\mathrm{L}(P) \subsetneq \mathrm{M}(P)$. *We write:* $\mathrm{L} \subsetneq \mathrm{M}$. *It is indicated by a filled arrow heads in Figure 1.*
3. L is incompatible with M, *if there exist polytopes* P, Q *such that* $\mathrm{M}(P) \not\subseteq \mathrm{L}(P)$ *and* $\mathrm{M}(Q) \not\subseteq \mathrm{L}(Q)$. *We write:* $\mathrm{L} \perp \mathrm{M}$. *It is indicated with an arrow with circle head and tail in Figure 1.*

In each of the above definitions, if either one of L *or* M *is defined only for polytopes* $P \subseteq [0,1]^n$, *then we confine the comparison to this class of polytopes.*

We establish the relations depicted in Figure 1 in the rest of the section.

3.1 Strength of ∂M for Arbitrary Admissible Cutting-Plane Procedures M

In order to show that ∂M refines M, we require the following technical lemma; see [8] for a similar result. We use the notation $\sigma_P(\cdot)$ to refer to the support function of a set P, i.e., $\sigma_P(c) = \sup\{cx \mid x \in P\}$.

Lemma 2. *Let $P, Q \subseteq \mathbb{R}^n$ be compact convex sets. If $\sigma_P(c) \leq \sigma_Q(c)$ for all $c \in \mathbb{Z}^n$, then $P \subseteq Q$.*

Theorem 2. *Let* M *be admissible. Then* $\partial \mathrm{M} \subseteq \mathrm{M}$.

Proof. Let P be a polytope. Since $\mathrm{M}(P) \subseteq P$ and $\partial\mathrm{M}(P) \subseteq P$, both $\mathrm{M}(P)$ and $\partial\mathrm{M}(P)$ are bounded. Moreover since $\mathrm{M}(P)$ is closed by definition, and $\partial\mathrm{M}(P)$ is defined as the intersection of halfspaces (thus a closed set), we obtain that $\mathrm{M}(P)$ and $\partial\mathrm{M}(P)$ are both compact convex sets. Thus, by Lemma 2, it is sufficient to compare the support functions of $\mathrm{M}(P)$ and $\partial\mathrm{M}(P)$ with respect to integer vectors only. Let $\sigma_{\mathrm{M}(P)}(c) = d$ for $c \in \mathbb{Z}^n$. We verify that $\sigma_{\partial\mathrm{M}(P)}(c) \leq \lfloor d \rfloor$. Observe that, $\mathrm{M}(P \cap \{cx \geq \lfloor d \rfloor + 1\}) \subseteq \mathrm{M}(P) \cap \{cx \geq \lfloor d \rfloor + 1\}$, where the inclusion follows from the inclusion preservation property of M. However note that since $cx \leq d$ is a valid inequality for $\mathrm{M}(P)$, we obtain that $\mathrm{M}(P) \cap \{cx \geq \lfloor d \rfloor + 1\} = \emptyset$. Thus, $\mathrm{M}(P \cap \{cx \geq \lfloor d \rfloor + 1\}) = \emptyset$ and so $cx \leq \lfloor d \rfloor$ is a valid inequality for $\partial\mathrm{M}(P)$. Equivalently we have $\sigma_{\partial\mathrm{M}(P)}(c) \leq \lfloor d \rfloor$, completing the proof. □

We next show that even if M is chosen arbitrarily, ∂M is at least as strong as the GC closure and the N_0 closure.

Theorem 3. *Let* M *be admissible. Then* $\partial \mathrm{M} \subseteq \mathrm{GC}$ *and* $\partial \mathrm{M} \subseteq \mathrm{N}_0$ *(the latter holding for polytopes $P \subseteq [0,1]^n$).*

Proof (Sketch of proof). The proof of $\partial \mathrm{M} \subseteq \mathrm{GC}$ is similar to the proof of Theorem 2 and we skip it here due to lack of space. Now let P be a polytope with $P \subseteq [0,1]^n$. For proving $\partial\mathrm{M}(P) \subseteq N_0(P)$, recall that $N_0 = \bigcap_{i \in [n]} P_i$ with $P_i := \mathrm{conv}\left((P \cap \{x_i = 0\}) \cup (P \cap \{x_i = 1\})\right)$. Therefore let $cx \leq d$ with $c \in \mathbb{Z}^n$ and $d \in \mathbb{Z}$ be valid for P_i with $i \in [n]$ arbitrary. In particular, $cx \leq d$ is valid for $P \cap \{x_i = l\}$ with $l \in \{0,1\}$. Thus we can conclude that $P \cap \{cx \geq d+1\} \cap \{x_i = l\} = \emptyset$ for $i \in \{0,1\}$. Therefore $x_i > 0$ and $x_i < 1$ are valid for $P \cap \{cx \geq d+1\}$ and so by Property 4 of Definition 1, $x_i \leq 0$ and $x_i \geq 1$ are valid $\mathrm{M}(P \cap \{cx \geq d+1\})$. We obtain $\mathrm{M}(P \cap \{cx \geq d+1\}) = \emptyset$ and thus $cx \leq d$ is valid for $\partial\mathrm{M}(P)$. □

3.2 Comparing M and ∂M for M Being GC, SC, N_0, N, or N_+

We now compare various closures and their associated $\mathbb{V}$-closures. The first result shows that the verification scheme of the Gomory-Chvátal procedure is at least as strong as split cuts.

Theorem 4. $\partial\mathrm{GC} \subseteq \mathrm{SC}$.

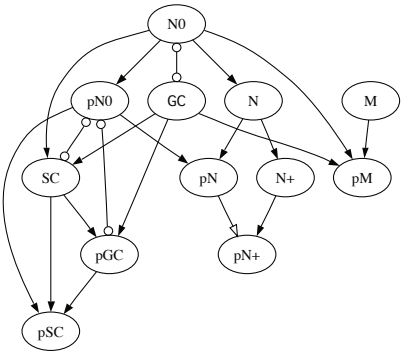


Fig. 1. Direct and $\mathbb{V}$-closures and their relations. pL in the figure represents ∂L and M is an arbitrarily weak admissible system.

Proof. Consider $cx \leq d$ being valid for $P \cap \{\pi x \leq \pi_0\}$ and $P \cap \{\pi x \geq \pi_0 + 1\}$ with $c, \pi \in \mathbb{Z}^n$ and $d, \pi_0 \in \mathbb{Z}$. Clearly, $cx \leq d$ is valid for $SC(P)$ and it suffices to consider inequalities $cx \leq d$ with this property; all others are dominated by positive combinations of these. Therefore consider $P \cap \{cx \geq d+1\}$. By $cx \leq d$ being valid for the disjunction $\pi x \leq \pi_0$ and $\pi x \geq \pi_0 + 1$ we obtain that $P \cap \{cx \geq d+1\} \cap \{\pi x \leq \pi_0\} = \emptyset$ and $P \cap \{cx \geq d+1\} \cap \{\pi x \geq \pi_0 + 1\} = \emptyset$. This implies that $P \cap \{cx \geq d+1\} \subseteq \{\pi x > \pi_0\}$ and similarly $P \cap \{cx \geq d+1\} \subseteq \{\pi x < \pi_0 + 1\}$. We thus obtain that $\pi x \geq \pi_0 + 1$ and $\pi x \leq \pi_0$ are valid for $\mathrm{GC}(P \cap \{cx \geq d+1\})$. It follows $\mathrm{GC}(P \cap \{cx \geq d+1\}) = \emptyset$. Thus $cx \leq d$ is valid for $\partial\mathrm{GC}(P)$. □

Next we compare $\mathbb{V}$-schemes of two closures that are comparable.

Lemma 3. *Let* L, M *be admissible such that* $\mathrm{L} \subseteq \mathrm{M}$. *Then* $\partial\mathrm{L} \subseteq \partial\mathrm{M}$.

In order to prove strict refinement or incompatibility between $\mathbb{V}$-closures the following lemma is helpful.

Proposition 1. *Let* L, M *be admissible. If* $P \subseteq [0,1]^n$ *is a polytope with* $P_I = \emptyset$ *such that* $\mathrm{M}(P) = \emptyset$ *and* $\mathrm{L}(P) \neq \emptyset$, *then* ∂L *does not refine* ∂M.

Proof (Sketch of proof). Let $G \subseteq [0,1]^n$ be a polytope. For $l \in \{0,1\}$, by $G_{x_{n+1}=l}$ we denote the polytope $S \subseteq [0,1]^{n+1}$ such that $S \cap \{x_{n+1} = l\} \cong G$ and S does not contain any other points. Consider the auxiliary polytope Q given as $Q := \mathrm{conv}\left(P_{x_{n+1}=1} \cup [0,1]^n_{x_{n+1}=0}\right)$. We next state a claim without proof due to lack of space: The inequality $x_{n+1} \leq 0$ is valid for $\partial L(Q)$ if and only if $L(Q \cap \{x_{n+1} \geq 1\}) = \emptyset$. Observe that $Q \cap \{x_{n+1} \geq 1\} \cong P$ and by assumption, we have $\mathrm{M}(P) = \emptyset$ but $\mathrm{L}(P) \neq \emptyset$ and therefore $\partial\mathrm{M}(Q) \not\supseteq \partial\mathrm{L}(Q)$. □

In the following lemmata, polytopes are presented that help establish the strict inclusion or incompatibility depicted in Figure 1, via Proposition 1.

Lemma 4. *$\partial N_0 \perp \partial\mathrm{GC}$ via $P_1 := \mathrm{conv}([0,1]^3 \cap \{x_1 + x_2 + x_3 = 3/2\}) \subseteq [0,1]^3$ and $P_2 := \mathrm{conv}(\{(\frac{1}{4}, \frac{1}{4}, 0), (\frac{1}{4}, \frac{1}{4}, 1), (\frac{1}{2}, 0, \frac{1}{2}), (\frac{1}{2}, 1, \frac{1}{2}), (0, \frac{1}{2}, \frac{1}{2}), (1, \frac{1}{2}, \frac{1}{2})\}) \subseteq [0,1]^3$.*

The proof of the next lemma uses Proposition 1 and a result from [7].

Lemma 5. *$\partial N_0 \perp \mathrm{SC}$ via $P_1 := A_3 \subseteq [0,1]^3$ and $P_2 := \mathrm{conv}([0,1]^3 \cap \{x_1 + x_2 + x_3 = 3/2\})$.*

A modified version of an example in [14] is used to verify the following result.

Lemma 6. *$\partial N \subsetneq \partial N_0$.*

4 Rank of Valid Inequalities with Respect to $\mathbb{V}$-Closures

In this section, we establish several bounds on the rank of $\partial\mathrm{M}$ for the case of polytopes $P \subseteq [0,1]^n$. Given a natural number k, we use the notation $\mathrm{M}^k(P)$ and $\mathrm{rk}_M(P)$ to be denote the k^{th} closure of P with respect to M and the rank of P with respect to M respectively. As $\partial\mathrm{M} \subseteq N_0$ we obtain the following result.

Theorem 5 (Upper bound in $[0,1]^n$). *Let M be admissible and $P \subseteq [0,1]^n$ be a polytope. Then $rk_{\partial\mathrm{M}}(P) \leq n$.*

Note that in general the property of M being admissible, does not guarantee that the upper bound on rank is n. For example, the GC closure can have a rank strictly higher than n (cf. [11,18]).

In quest for lower bounds on the rank of 0/1 polytopes, we note that among polytopes $P \subseteq [0,1]^n$ that have $P_I = \emptyset$, the polytope $A_n = \{x \in [0,1]^n \mid \sum_{i \in I} x_i + \sum_{i \notin I}(1 - x_i) \geq \frac{1}{2} \quad \forall\, I \subseteq [n]\}$ has maximal rank (of n) for many admissible systems [16]. We will now establish that $\partial\mathrm{M}$ is not unrealistically strong by showing that it is subject to similar limitations. Recall that we do not require *short verification* (property (7)) for $\partial(\mathrm{M})$ which is the basis for the lower bound in [17, Corollary 23] for admissible systems. We will show that the lower bound for $\partial\mathrm{M}$ is *inherited* from the original operator M. Let

$$F_n^k := \{x \in \{0, 1/2, 1\}^n \mid \text{ exactly } k \text{ entries equal to } 1/2\},$$

and let $A_n^k := \mathrm{conv}\left(F_n^k\right)$ be the convex hull of F_n^k. (Note $A_n^1 = A_n$.) With F being a face of $[0,1]^n$ let $I(F)$ denote the index set of those coordinate that are fixed by F.

Lemma 7. *Let M be admissible and let $\ell \in \mathrm{N}$ such that $A_n^{k+\ell} \subseteq \mathrm{M}(A_n^k)$ for all $n, k \in \mathrm{N}$ with $k + \ell \leq n$. If $n \geq k + 2\ell + 1$, then $A_n^{k+2\ell+1} \subseteq \partial\mathrm{M}(A_n^k)$.*

Proof. Let $P := A_n^k$ and let $cx \leq d$ with $c \in \mathbb{Z}^n$ and $d \in \mathbb{Z}$ be valid for $\partial\mathrm{M}(P)$. Without loss of generality we assume $\mathrm{M}(P \cap \{cx \geq d + 1\}) = \emptyset$, i.e., $cx \leq d$ is one of the defining inequalities. We claim that

$$A_{k+\ell}^k \cong A_n^k \cap F \not\subseteq P \cap \{cx \geq d + 1\} \tag{6}$$

for all $(k+\ell)$-dimensional faces F of $[0,1]^n$. Assume by contradiction that $A_n^k \cap F \subseteq P \cap \{cx \geq d+1\}$. As $A_{k+\ell}^{k+\ell} \subseteq \mathrm{M}(A_{k+\ell}^k)$ by assumption we obtain $\emptyset \neq A_{k+\ell}^{k+\ell} \subseteq \mathrm{M}(A_{k+\ell}^k) \subseteq \mathrm{M}(P \cap \{cx \geq d+1\}$ which contradicts the validity of $cx \leq d$ over $\partial\mathrm{M}(P)$.

Without loss of generality we can further assume that $c \geq 0$ and $c_i \geq c_j$ whenever $i \leq j$ by applying coordinate flips and permutations.

Next we claim that for all $(k+\ell)$-dimensional faces F of $[0,1]^n$ the point v^F defined as

$$v_i^F := \begin{cases} \in \{0,1\} \text{ according to } F, \text{ for all } i \in I(F) \\ 0, \text{ if } c_i \text{ is one of the } \ell \text{ largest coefficients of } c \text{ with } i \notin I(F) \\ 1/2, \text{ otherwise} \end{cases} \tag{7}$$

for $i \in [n]$ must not be contained in $P \cap \{cx \geq d+1\}$, i.e., $cv^F < d+1$ and so $cv^F \leq d+1/2$. Note that $v^F \in P$ and observe that $v^F := \mathrm{argmin}_{x \in F_n^k \cap F} cx$. Therefore, if $v^F \in P \cap \{cx \geq d+1\}$, then $A_n^k \cap F \subseteq P \cap \{cx \geq d+1\}$ which in turn contradicts (6). This claim holds in particular for those faces F fixing coordinates to 1.

Finally, we claim that $A_n^{k+2\ell+1} \subseteq P \cap \{cx \leq d\}$. It suffices to show that $cv \leq d$ for all $v \in F_n^{k+2\ell+1}$ and we can confine ourselves to the worst case v given by

$$v_i := \begin{cases} 1, \text{ if } i \in [n-(k+2\ell+1)] \\ 1/2, \text{ otherwise.} \end{cases}$$

Observe that $cv \geq cw$ holds for all $w \in F_n^{k+2\ell+1}$. Let F be the $(k+\ell)$-dimensional face of $[0,1]^n$ obtained by fixing the first $n-(k+\ell)$ coordinates to 1. Then

$$\begin{aligned} cv &= \sum_{i=1}^{n-(k+2\ell+1)} c_i + \frac{1}{2} \sum_{i=n-(k+2\ell+1)+1}^{n} c_i \\ &\leq \sum_{i=1}^{n-(k+\ell)} c_i - \frac{1}{2} c_{n-(k+\ell)} + \sum_{i=n-(k+\ell)+1}^{n-k} 0 + \frac{1}{2} \sum_{i=(n-k)+1}^{n} c_i \\ &= cv^F - \frac{1}{2} c_{n-(k+\ell)} \leq d + \frac{1}{2} - \frac{1}{2} c_{n-(k+\ell)}. \end{aligned}$$

If $c_{n-(k+\ell)} \geq 1$, then $cv \leq d$. Therefore consider the case $c_{n-(k+\ell)} = 0$. Then we have that $c_i = 0$ for all $i \geq n-(k+\ell)$. In this case cv^F is integral and $cv^F < d+1$ implies $cv^F \leq d$. So $cv \leq cv^F \leq d$ follows, which completes the proof. □

Theorem 6 (Lower bound for A_n). *Let* M *be admissible and let* $\ell \in \mathrm{N}$ *such that* $A_n^{k+\ell} \subseteq \mathrm{M}(A_n^k)$ *for all* $n, k \in \mathrm{N}$ *with* $k+\ell \leq n$. *If* $n \geq k+2\ell+1$, *then* $rk_{\partial\mathrm{M}}(A_n) \geq \left\lfloor \frac{n-1}{2\ell+1} \right\rfloor$.

Proof. We will show the $A_n^{1+k(2\ell+1)} \subseteq (\partial\mathrm{M})^k(A_n)$ as long as $n \geq k+2\ell+1$. The proof is by induction on k. Let $k=1$, then $A_n^{1+2\ell+1} \subseteq \partial\mathrm{M}(A_n^1) = \partial\mathrm{M}(A_n)$ by

Lemma 7. Therefore consider $k > 1$. Now $(\partial \mathrm{M})^k(A_n) = \partial \mathrm{M}((\partial \mathrm{M})^{k-1}(A_n)) \supseteq \partial \mathrm{M}(A_n^{1+(k-1)(2\ell+1)}) \supseteq A_n^{1+k(2\ell+1)}$, where the first inclusion follows by induction and the second by Lemma 7 again. Thus $(\partial \mathrm{M})^k(A_n) \neq \emptyset$ as long as $1 + k(2\ell + 1) \leq n$, which is the case as long as $k \leq \left\lfloor \frac{n-1}{2\ell+1} \right\rfloor$ and we can thus conclude $\mathrm{rk}_{\partial \mathrm{M}}(A_n) \geq \left\lfloor \frac{n-1}{2\ell+1} \right\rfloor$. □

For $\mathrm{M} \in \{\mathrm{GC}, \mathrm{SC}, N_0, N, N_+\}$ we have that $\ell = 1$ (see e.g., [17]) and therefore we obtain the following corollary.

Corollary 1. *Let* $\mathrm{M} \in \{\mathrm{GC}, N_0, N, N_+, \mathrm{SC}\}$ *and* $n \in \mathrm{N}$ *with* $n \geq 4$. *Then* $rk_{\partial \mathrm{M}}(A_n) \geq \left\lfloor \frac{n-1}{3} \right\rfloor$.

We can also derive an upper bound on the rank of A_n which is a consequence of [17, Lemma 5].

Lemma 8 (Upper bound for A_n). *Let* M *be admissible and* $n \in \mathrm{N}$. *Then* $rk_{\partial \mathrm{M}}(A_n) \leq n - 2$.

5 $\mathbb{V}$-Closures for Well-Known and Structured Problems

We first establish a useful lemma which holds for any $\partial \mathrm{M}$ with M being admissible. The lemma is analogous to Lemma 1.5 in [14].

Lemma 9. *Let* M *be admissible and let* $P \subseteq [0,1]^n$ *be a polytope with* $(c,d) \in \mathbb{Z}_+^{n+1}$. *If* $cx \leq d$ *is valid for* $P \cap \{x_i = 1\}$ *for every* $i \in [n]$ *with* $c_i > 0$, *then* $cx \leq d$ *is valid for* $\partial \mathrm{M}(P)$.

Proof. Clearly, $cx \leq d$ is valid for P_I; if $x \in P \cap \mathbb{Z}^n$ non-zero, then there exists an $i \in [n]$ with $x_i = 1$, otherwise $cx \leq d$ is trivially satisfied. We claim that $cx \leq d$ is valid for $\partial \mathrm{M}$. Let $Q := P \cap \{cx \geq d + 1\}$ and observe that $Q \cap \{x_i = 1\} = \emptyset$ for any $i \in [n]$ with $c_i > 0$. Therefore by coordinating rounding $\mathrm{M}(Q) \subseteq \bigcap_{i \in [n]: c_i > 0} \{x_i = 0\}$. By definition of Q we also have that $\mathrm{M}(Q) \subseteq \{cx \geq d + 1\}$. Since $c \geq 0$ and $d \geq 0$ we deduce $\mathrm{M}(Q) = \emptyset$ and the claim follows. □

5.1 Monotone Polytopes

The following theorem is a direct consequence of Lemma 9 and follows in a similar fashion as Lemma 2.7 in [5] or Lemma 2.14 in [14].

Theorem 7. *Let* M *be admissible. Further, let* $P \subseteq [0,1]^n$ *be a polytope and* $(c,d) \in \mathbb{Z}_+^{n+1}$ *such that* $cx \leq d$ *is valid for* $P \cap F$ *whenever* F *is an* $(n-k)$*-dimensional face of* $[0,1]^n$ *obtained by fixing coordinates to* 1. *Then* $cx \leq d$ *is valid* $(\partial \mathrm{M})^k(P)$.

We call a polytope $P \subseteq [0,1]^n$ *monotone* if $x \in P$, $y \in [0,1]^n$, and $y \leq x$ (coordinate-wise) implies $y \in P$. We can derive the following corollary from Theorem 7 which is the analog to Lemma 2.7 in [5].

Corollary 2. *Let* M *be admissible and let* $P \subseteq [0,1]^n$ *be a monotone polytope with* $\max_{x \in P_I} ex = k$. *Then* $rk_{\partial \mathrm{M}}(P) \leq k+1$.

Proof. Observe that since P is monotone, so is P_I and thus P_I possesses an inequality description $P = \{x \in [0,1]^n \mid Ax \leq b\}$ with $A \in \mathbb{Z}_+^{m \times n}$ and $b \in \mathbb{Z}_+^n$ for some $m \in \mathrm{N}$. Therefore it suffices to consider inequalities $cx \leq d$ valid for P_I with $c, d \geq 0$. As $\max_{x \in P_I} ex = k$ and P is monotone, we claim that $P \cap F = \emptyset$ whenever F is an $n-(k+1)$ dimensional face of $[0,1]^n$ obtained by fixing $k+1$ coordinates to 1. Assume by contradiction that $x \in P \cap F \neq \emptyset$. As $P \cap F$ is monotone, the point obtained by setting all fractional entries of x to 0 is contained in $P_I \cap F$ which is a contradiction to $\max_{x \in P_I} ex = k$. Therefore $cx \leq d$ is valid for all $P \cap F$ with F being an $n-(k+1)$ dimensional face of $[0,1]^n$ obtained by fixing $k+1$ coordinates to 1. The result follows now by using Theorem 7. □

5.2 Stable Set Polytope

Given a graph $G := (V, E)$, the fractional stable set polytope of G is given by $\mathrm{FSTAB}(G) := \{x \in [0,1]^n \mid x_u + x_v \leq 1 \ \forall (u,v) \in E\}$. Now Lemma 9 can be used to prove the following result.

Theorem 8. *Clique Inequalities, odd hole inequalities, odd anti-hole inequalities, and odd wheel inequalities are valid for* $\partial \mathrm{M}(FSTAB(G))$ *with* M *being an admissible operator.*

5.3 The Traveling Salesman Problem

So far we have seen that transitioning from a general cutting-plane procedure M to its $\mathbb{V}$-scheme ∂M can result in a significantly lower rank for valid inequalities, potentially making them accessible in a small number of rounds. However, we will now show that the rank of (the subtour elimination relaxation of) the traveling salesman polytope remains high, even when using $\mathbb{V}$-schemes of strong operators such as SC or N_+. For $n \in \mathbb{N}$, let $G = (V, E)$ be the complete graph on n vertices and $H_n \subseteq [0,1]^n$ be the polytope given by (see [5] for more details)

$$\begin{aligned} x(\delta(\{v\})) &= 2 && \forall\ v \in V \\ x(\delta(W)) &\geq 2 && \forall\ \emptyset \subsetneq W \subsetneq V \\ x_e &\in [0,1] && \forall e \in E. \end{aligned}$$

Note that the (ambient) dimension of H_n is $\Theta(n^2)$. We obtain the following statement which is the analog to [5, Theorem 4.1]. A similar result for the admissible systems M in general can be found in full-length version of [17].

Theorem 9. *Let* $\mathrm{M} \in \{\mathrm{GC}, N_0, N, N_+, \mathrm{SC}\}$. *For* $n \in \mathbb{N}$ *and* H_n *as defined above we have* $rk_{\partial \mathrm{M}}(H_n) \in \Theta(n)$. *In particular* $rk_{\partial \mathrm{M}}(H_n) \in \Theta(\sqrt{dim(P)})$.

Proof (Sketch of proof). As shown in [3] or [5, Theorem 4.1] H_n contains a copy of $A_{\lfloor n/8 \rfloor}$. The lower bound follows from Corollary 1 and the upper bound follows from Corollary 2 as shown in [5]. □

The same result can be shown to hold for the asymmetric TSP problem (see [3] and [5]).

5.4 General Polytopes in $\mathbb{R}^2$

The GC rank of valid inequalities for polytopes in $\mathbb{R}^2$ can be arbitrarily high; see example in [15]. However, ∂GC is significantly stronger as shown next.

Theorem 10. *Let P be a polytope in $\mathbb{R}^2$. Then $\partial\mathrm{GC}(P) = P_I$.*

Proof (Sketch of proof). The proof is divided into various cases based on the dimension of P_I. Here we only present the proof for the case when $\dim(P_I) = 2$. In this case, every facet-defining inequality $cx \leq d$ satisfies at least two integer points belonging to P_I. Let $Q := P \cap \{x \in \mathbb{R}^2 \mid cx \geq d\}$. Then observe that: (i) Q is a lattice-free polytope; (ii) exactly one side of Q contains multiple integer points. This is the side of Q given by the inequality $cx \geq d$. Other sides of Q contain no integer point. Let T be a maximal lattice-free convex set containing Q. By (ii), $cx \geq d$ defines a face of T that contains two or more integer points. Therefore T is a type 1 or type 2 maximal lattice-free triangle; see [10]. Since T is a triangle of type 1 or type 2, it is contained in two sets of the form $\{\pi_0^1 \leq \pi^1 x \leq \pi_0^1 + 1\}$ and $\{\pi_0^2 \leq \pi^2 x \leq \pi_0^2 + 1\}$ where $\pi^1, \pi^2 \in \mathbb{Z}^2$ and $\pi_0^1, \pi_0^2 \in \mathbb{Z}$; see [9]. Moreover π^1 and π^2 can be selected such that $\pi^1 = c$, $\pi_0^1 = d$ and the two integer points x^1 and x^2 belonging to P and satisfying $cx = d$ satisfy $\pi^2 x^1 = \pi_0^2$ and $\pi^2 x^2 = \pi_0^2 + 1$. Therefore $Q \cap \{cx \geq d + 1\} \subseteq T \cap \{cx \geq d + 1\} \subseteq \{\pi_0^2 \leq \pi^2 x \leq \pi_0^2 + 1\}$. Moreover, since the integer points belonging to the boundary of Q satisfy the condition $cx = d$, we obtain that integer points that satisfy $cx \geq d + 1$ and lie on the boundary of the set $\{\pi_0^2 \leq \pi^2 x \leq \pi_0^2 + 1\}$ do not belong to Q. Now by using convexity of Q and the location of integer points in $P \cap \{cx = d\}$, we can verify that $Q \cap \{cx \geq d + 1\}$ lies in the interior of the set $\{\pi_0^2 \leq \pi^2 x \leq \pi_0^2 + 1\}$. Therefore $\mathrm{GC}(Q \cap \{cx \geq d + 1\}) = \emptyset$. However, since $Q \cap \{cx \geq d+1\} = P \cap \{cx \geq d+1\}$, we can obtain the facet-defining inequality $cx \leq d$ using the ∂GC operator applied to P. □

6 Concluding Remarks

In this paper, we consider a new paradigm for generating cutting-planes. Rather than *computing* a cutting-plane we suppose that the cutting-plane is given, either by a *deliberate construction* or guessed in some other way and then we *verify* its validity using a regular cutting-plane procedure. We have shown that cutting-planes obtained via the verification scheme can be very strong, significantly exceeding the capabilities of the regular cutting-plane procedure. This superior strength is illustrated, for example, in Theorem 2, Theorem 4, Figure 1,

Theorem 5, Lemma 8, Theorem 7, Theorem 8, Theorem 9 and Theorem 10. On the other hand, we also show that the verification scheme is not unrealistically strong, as illustrated by Theorem 6 and Theorem 9.

References

1. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed integer 0-1 programs. Mathematical Programming 58, 295–324 (1993)
2. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Mathematics 4, 305–337 (1973)
3. Chvátal, V., Cook, W., Hartmann, M.: On cutting-plane proofs in combinatorial optimization. Linear Algebra and its Applications 114, 455–499 (1989)
4. Cook, W., Coullard, C.R., Turan, G.: On the complexity of cutting plane proof. Mathematical Programming 47, 11–18 (1990)
5. Cook, W., Dash, S.: On the matrix cut rank of polyhedra. Mathematics of Operations Research 26, 19–30 (2001)
6. Cook, W., Kannan, R., Schrijver, A.: Chvátal closures for mixed integer programming problems. Mathematical Programming 58, 155–174 (1990)
7. Cornuéjols, G., Li, Y.: On the rank of mixed 0-1 polyhedra. Mathematical Programming 91, 391–397 (2002)
8. Dadush, D., Dey, S.S., Vielma, J.P.: The Chvátal-Gomory Closure of Strictly Convex Body (2010), `http://www.optimization-online.org/DB_HTML/2010/05/2608.html`
9. Dash, S., Dey, S.S., Günlük, O.: Two dimensional lattice-free cuts and asymmetric disjunctions for mixed-integer polyhedra (2010), `http://www.optimization-online.org/DB_HTML/2010/03/2582.html`
10. Dey, S.S., Wolsey, L.A.: Two row mixed integer cuts via lifting. Mathematical Programming 124, 143–174 (2010)
11. Eisenbrand, F., Schulz, A.S.: Bounds on the Chvátal rank of polytopes in the 0/1-cube. Combinatorica 23, 245–262 (2003)
12. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bulletin of the American Mathematical Society 64, 275–278 (1958)
13. Kelley, J.E.: The cutting plane method for solving convex programs. Journal of the SIAM 8, 703–712 (1960)
14. Lovász, L., Schrijver, A.: Cones of matrices and set-functions and 0-1 optimization. SIAM Journal on Optimization 1, 166–190 (1991)
15. Nemhauser, G.L., Wolsey, L.A.: Integer and combinatorial optimization. Wiley Interscience, Hoboken (1988)
16. Pokutta, S., Schulz, A.S.: Characterization of integer-free 0/1 polytopes with maximal rank, working paper
17. Pokutta, S., Schulz, A.S.: On the rank of generic cutting-plane proof systems. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 450–463. Springer, Heidelberg (2010)
18. Pokutta, S., Stauffer, G.: A new lower bound technique for the Gomory-Chvátal procedure (2010), `http://www.optimization-online.org/DB_HTML/2010/09/2748.html`

Contact Center Scheduling with Strict Resource Requirements

Aman Dhesi[1], Pranav Gupta[2], Amit Kumar[3],
Gyana R. Parija[2], and Sambuddha Roy[2]

[1] Department of Computer Science, Princeton University
adhesi@princeton.edu
[2] IBM Research – India, New Delhi
{prguptan,gyana.parija,sambuddha}@in.ibm.com
[3] Department of Computer Science and Engg.,
Indian Institute of Technology, Delhi
amitk@cse.iitd.ac.in

Abstract. Consider the following problem which often arises in contact center scheduling scenarios. We are given a set of employees where each employee can be deployed for shifts consisting of L consecutive time units. Further, each employee specifies a set of possible start times, and can be deployed for a bounded number of shifts only. At each point of time t, we are also given a lower bound r_t on the number of employees that should be present at this time. The goal is to find a schedule for the employees such that the number of time slots whose requirements are met is maximized. Such problems naturally arise in many other situations, e.g., sensor networks and cloud computing.

The strict nature of the resource requirement makes this problem very hard to approximate. In this paper, we give a bicriteria approximation algorithm for this problem. Given a parameter $\varepsilon > 0$, we give an $O(\frac{1}{\varepsilon^3} \cdot \log \frac{1}{\varepsilon})$-approximation algorithm for this problem, where we count those time slots for which we satisfy at least $(1-\varepsilon)$-fraction of the requirement. Our techniques involve a configuration LP relaxation for this problem, and we use non-trivial structural properties of an optimal solution to solve this LP relaxation. We even consider the more general problem where shift lengths of different employees can vary significantly. In this case, we show that even finding a good bicriteria approximation is hard (under standard complexity theoretic assumptions).

1 Introduction

Scheduling employees is a central problem in workforce management in contact centers [ICWC10, FHF+02]. Typically, in these settings, we have a reasonably accurate forecast of demands which will arrive at any point of time (also referred as *time slots*) in future. This in turn implies that we roughly know, for each time slot, how many employees will be needed. The employees are usually required to work for shifts of a certain fixed duration, which we shall denote by L. Many of these employees work as part-time workers and often have varying constraints on

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 156–169, 2011.

their availability. In other words, for each employee, we know at what times they can possibly start on a shift. Given these constraints, we would like to schedule these employees such that we maximize the number of timeslots for which we have the requisite number of employees. The setting of contact center often demands, via service level agreements (SLA's), that we satisfy the requirements of a large fraction of timeslots.

The framework that we consider in this paper is quite general and can capture many other situations :

- Sensor networks : Sensors have limited power and can last for only a certain duration. In many cases, we would like to use a sensor non-preemptively for its entire lifetime. Suppose we are given, for each time unit, how many sensors we would like to be active at that time. The problem is now to schedule the sensors (i.e., decide when to turn them on) such that we maximize the number of timeslots whose requirements are met.
- Cloud computing : In cloud computing, unit size tasks arrive over a period of time, and we process these by assigning them to a set of processors available over a network. Various constraints over availability of a processor may dictate that we can only use them in one (or several) possible slots for a certain period of time.
- Energy Management: We are given the energy requirement for each point of time in future, and are also given a set of available energy sources. However, each energy source can run for only a fixed duration of time, and may be available at some specific timeslots only.

We now define the problem more concretely. As mentioned earlier, we divide time into slots : let T denote the set of time-slots. We shall often refer the employees as *resources* – let R denote the set of resources. For each time $t \in T$, we are given a requirement r_t, and a profit p_t. For each resource $i \in R$, we are given a parameter α_i and a set of intervals $\mathcal{E}_i$, where each interval in $\mathcal{E}_i$ is of uniform length L. We would like to select at most α_i intervals from $\mathcal{E}_i$ for each i such that the number of *satisfied* timeslots is maximized. A timeslot is satisfied if there are at least r_t selected intervals, no two from the same set $\mathcal{E}_i$, which contain t. We call this problem the `MaxStaff` problem.

Related Work. There has been much work on workforce management in the operations research community [AAM07, GKM03]. Contact center scheduling presents new kinds of constraints where employee availability (and skills) have very high variance [Rey03]. To the best of our knowledge, ours is the first work to view these problems from an approximation algorithms perspective.

The `MaxStaff` problem is a mixed packing-covering problem – the constraints on resources are of packing type, whereas we need to cover the demands. If we relax the strict nature of the requirements, i.e., if we are just interested in maximizing the sum over all timeslots of the *fraction* to which it is satisfied, then the problem becomes a packing problem. In fact, in this case, it falls under the framework of maximizing a sub-modular function under matroid and knapsack constraints [CCPV07, GRST10, LMNS09].

Our Contributions. Our first result is that the `MaxStaff` problem is as hard to approximate as the `Maximum Independent Set` problem.

Theorem 1. *There is an approximation preserving reduction from the* `Maximum Independent Set` *problem to the* `MaxStaff` *problem.*

Since very strong hardness results are known for the `Maximum Independent Set` problem [ST00], we look for bicriteria approximation algorithms of the following kind. Given an instance $\mathcal{I}$ of the `MaxStaff` problem, let $\texttt{opt}(\mathcal{I})$ denote the cost of the optimum solution for $\mathcal{I}$. For a parameter γ, $0 \leq \gamma \leq 1$, we say that a solution γ-satisfies a timeslot t if there are at least $\gamma \cdot r_t$ selected intervals containing t, no two from the same set $\mathcal{E}_i$ for any i. Given parameters α, γ, $0 \leq \gamma \leq 1$, we say that a solution is (α, γ)-bicriteria approximate if the total profit of timeslots which are γ-satisfied by this solution is at least $\alpha \cdot \texttt{opt}(\mathcal{I})$. An (α, γ)-bicriteria algorithm is one which produces solutions with this property. We show that for any constant ε, we can get a $(poly(\varepsilon), 1-\varepsilon)$-bicriteria approximation algorithm for this problem.

Theorem 2. *For any constant $\varepsilon > 0$, there is a poly-time $\left(O\left(\frac{\varepsilon^3}{\log \frac{1}{\varepsilon}}\right), 1-\varepsilon\right)$-bicriteria approximation algorithm for the* `MaxStaff` *problem.*

We then consider the case when the shift length L may vary over different elements in R, i.e., if the intervals in $\cup_{i \in R} \mathcal{E}_i$ can be of arbitrary lengths. On the positive side, the proof of Theorem 2 can be easily extended to show that if the ratio of the maximum length to the minimum length of an interval is β, then one can get an $\left(O\left(\frac{\varepsilon^3 \cdot \beta}{\log \frac{1}{\varepsilon}}\right), 1-\varepsilon\right)$-bicriteria approximation algorithm. However, we show that in this case, one cannot hope for a significantly better result.

Theorem 3. *For any small enough (but constant) $\varepsilon > 0$, we cannot have a poly-time $\left(O\left(2^{c\sqrt{\log N}}\right), 1-\varepsilon\right)$-bicriteria approximation algorithm for the* `MaxStaff` *problem, unless $NP \subseteq DTIME\left(n^{O(\log n)}\right)$. Here N denotes the size of the input instance, and c is a constant (depending on ε).*

Our Techniques. One can write a "natural" LP relaxation for the `MaxStaff` problem. However, this turns out to have unbounded gap. We first show that if we are willing to lose a constant factor (depending on ε) in the profit, then there is a significant amount of structure in any solution. We first consider the special case when all the requirements are same : this happens to capture many of the difficulties. First, we divide the timeline into *blocks* : each block is a sequence of L continuous time slots. Then we show that, with a factor 2 loss in approximation, we can assume that each selected interval contributes to satisfying the requirements of only one block. Thus, we can decompose a solution into several independent solutions, one for each block (of course, we must satisfy the global requirement that we pick only α_i intervals from $\mathcal{E}_i$). We then focus on a particular block, and think of the solution restricted to just this block. We further simplify the structure of a solution : we show that we can find a continuous set of timeslots in a block, which we call a sub-block, and select a subset of

intervals in the solution such that the following conditions hold : (i) each interval in the latter solution contains the entire sub-block and satisfy the requirements of the timeslots in this sub-block (upto an ε factor loss in the requirement), and (ii) the total profit accrued from this sub-block is at least a constant times the total profit obtained by the solution from the entire block. This simple structure allows us to write a configuration LP – we show that we can solve this LP and a simple randomized rounding based algorithm gives a good approximation algorithm. The extension to the case of arbitrary requirements is based on standard geometric grouping ideas.

For the hardness result, the reduction from the `Maximum Independent Set` problem follows in a straight-forward manner. Using this one can also show that if we want a bicriteria approximation for a parameter ε, then there is a constant (depending on ε) hardness. For the case when the lengths of the intervals in the input can vary (significantly), we show that this construction can be recursively composed to amplify the hardness of the problem to that mentioned in Theorem 3.

Organization of the Paper. In Section 2, we define the problem formally and give some related definitions. In Section 3, we prove both the hardness results. In Section 4, we describe the bi-criteria approximation algorithm. In Section 4.1, we first consider the case when all the requirements r_t are same. This special case captures many of the ideas, and simplifies our presentation. In Section 4.2, we outline the ideas needed to extend this to the case of arbitrary requirements. The details of the algorithm for the latter case are deferred to the full version.

2 Problem Definition and Notation

We define the `MaxStaff` problem. An input instance $\mathcal{I}$ is specified by a tuple $(T, R, \mathcal{E}, r, \alpha, p, L)$. The elements of T correspond to consecutive time slots, and they are numbered in this order from 1 to $|T|$. The set R is the set of available resources, $\mathcal{E}$ is a mapping from R to the power set of all intervals of length L in T. We denote by $\mathcal{E}_i$ the set of intervals (of length L) associated with $i \in R$. The quantities r, p, α are mappings from the sets T, T, R to the set of positive integers respectively. The value r_t denotes the requirement of timeslot t, p_t denotes its profit, and α_i denotes the "availability" of resource i, i.e., the maximum number of intervals we may pick from the set $\mathcal{E}_i$.

We say that two intervals I, I' in the input are *mutually distinct* if they belong to different sets $\mathcal{E}_i$. A solution $\mathcal{S}$ selects for each $i \in R$, a subset $\mathcal{E}_i(\mathcal{S})$ of at most α_i intervals from $\mathcal{E}_i$. Let $\mathcal{E}(\mathcal{S})$ denote $\cup_i \mathcal{E}_i(\mathcal{S})$. We say that $t \in T$ gets *satisfied* by $\mathcal{S}$ if there are at least r_t mutually distinct intervals containing t in the set $\mathcal{E}(S)$. Note that the intervals in $\mathcal{E}_i(\mathcal{S})$ can overlap, but they contribute at most once towards the number of mutually distinct intervals containing any timeslot. The goal is to maximize the total profit of time slots which get satisfied.

For the hardness result, the length L may vary with the resource i – for each $i \in R$, the input instance specifies a length L_i. So all intervals in $\mathcal{E}_i$ are of length L_i.

3 Hardness Results

3.1 Proof of Theorem 1

We reduce the `Maximum Independent Set` problem to the `MaxStaff` problem. Consider an instance $\mathcal{I}' = (G = (V, E))$ for the `Maximum Independent Set` problem. We construct an instance $\mathcal{I} = (T, R, \mathcal{E}, r, \alpha, p, L)$ of the `MaxStaff` problem as follows. Let $V = \{v_1, \ldots, v_n\}$. The set T will be equal to $\{1, \ldots, |V|\}$ and a timeslot $t \in T$ will correspond to the vertex v_t. For each edge $e \in E$, we have an element $i_e \in R$. We now describe the set $\mathcal{E}$ in $\mathcal{I}$. For an edge $e = (v_a, v_b) \in E$, the set $\mathcal{E}_{i_e}$ contains two intervals – $\{a\}, \{b\}$. Note that the parameter L is equal to 1. For every $t \in T$, the requirement r_t is equal to the degree of the vertex v_t in G. Further, we have $\alpha_i = 1$ for all $i \in R$, and $p_t = 1$ for all $t \in T$ (see Figure 1 for an example).

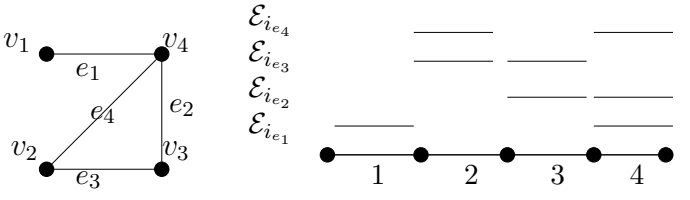


Fig. 1. Example showing the reduction from `Maximum Independent Set` to the `MaxStaff` problem

Claim. There is a solution to the `MaxStaff` problem instance $\mathcal{I}$ of profit at least k iff there is an independent set in G' of size k.

Proof. Suppose there is an independent set S of size k in G. Let $T' = \{t : v_t \in S\}$ be the corresponding timeslots in T (in the instance $\mathcal{I}$). Consider the following solution : for any edge $e = (v_a, v_b) \in E$, at most one of the vertices in v_a and v_b belongs to S – assume wlog that $v_a \in S$. Then we pick the interval $\{a\} \in \mathcal{E}_{i_e}$ (in case none of these vertices is in S, we pick any one of the two intervals in $\mathcal{E}_{i_e}$). It is easy to check that this solution satisfies the requirements of all the timeslots in T'.

Conversely, suppose that there is a solution for $\mathcal{I}$ which satisfies k timeslots in T – call this set T'. Again, it is easy to check that the set $S = \{v_t : t \in T'\}$ is an independent set in G. □

This concludes the proof of Theorem 1.

3.2 Proof of Theorem 3

Fix a small enough constant $\varepsilon > 0$. Assume wlog that $\frac{1}{\varepsilon}$ is an integer and let Δ denote this quantity. We use the following result of Samorodnitsky and Trevisan [ST00] which shows hardness of the `Maximum Independent Set` problem on bounded degree graphs.

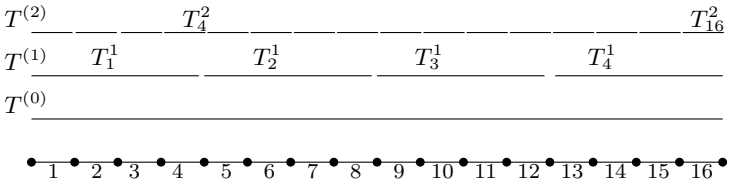


Fig. 2. The sets of intervals in $T^{(0)}$ and $T^{(1)}$ for the graph in Figure 1

Theorem 4. *[ST00] Unless P=NP, there is no poly-time* $\frac{\Delta}{2^{O(\sqrt{\log \Delta})}}$*-approximation algorithm for the* `Maximum Independent Set` *problem on graphs of bounded degree* Δ*.*

We start with an instance $\mathcal{I}' = (G = (V, E))$ of the `Maximum Independent Set`-problem, where the maximum degree of a vertex in G is Δ. For ease of notation, we can in fact assume by adding self-loops that the degree of each vertex is exactly Δ – we modify the definition of an independent set as a set of vertices such that for any edge, at most *one* of its end-points is in this set.

We now construct an instance $\mathcal{I} = (T, R, \mathcal{E}, r, \alpha, p, L)$ of the `MaxStaff` problem. Recall that L_i denotes the length of the intervals in $\mathcal{E}_i$. The idea behind the proof is the following : we start with the reduction from `Maximum Independent Set` to `MaxStaff` as described in the previous section. Now we stretch each timeslot n times, i.e., we replace it by n consecutive timeslots (so the intervals which were earlier of length 1 now have length n). For each of these newly created n timeslots (corresponding to one timeslot in the original instance), we again embed the `MaxStaff` instance used in the reduction. This process is continued recursively for K stages. We now give details of our construction.

The set T will have size n^K, where $n = |V|$ in the instance $\mathcal{I}'$ and K is a parameter to be chosen later. We shall construct the set of elements in R and the set $\mathcal{E}$ in several stages. We first divide the timeline T into a laminar family of sets. For each value of k between 0 and K, we do the following : divide the set T (starting from the first timeslot) into disjoint intervals of n^{K-k} consecutive points each – call these intervals $T^k_1, \ldots, T^k_{r_k}$, where $r_k = n^k$. Let $T^{(k)}$ denote the set of these intervals. It is easy to check that T^k_r contains n intervals from the set $T^{(k+1)}$ and is disjoint from other intervals in $T^{(k+1)}$ (see Figure 2 for an example with $K = 2$).

We now construct the sets R and $\mathcal{E}$. For each value of k, $0 \leq k \leq K-1$, and $1 \leq r \leq n^k$, we add a set of elements $R(k, r)$ to the set R. Recall that T^k_r contains n intervals from the set $T^{(k+1)}$ – call these intervals $T^{k+1}_{s_1}, \ldots, T^{k+1}_{s_n}$. Let the vertices in V be $v_1, \ldots, v_n$ (recall that V is the set of vertices in the `Maximum Independent Set` instance $\mathcal{I}'$). For each edge $e = (v_a, v_b) \in E$, we add an element $e(k, r)$ to $R(k, r)$. Further, we have two intervals in $\mathcal{E}_{e(k,r)}$: $T^{k+1}_{s_a}$ and $T^{k+1}_{s_b}$ – we denote these intervals by $I^1(e(k, r))$ and $I^2(e(k, r))$. Notice that there are exactly Δ intervals of the form $I^l(e(k, r))$, $l = 1, 2$, $e \in E$, which are equal to $T^{(k+1)}_{s_a}$, $1 \leq a \leq n$. Note that $L_i = n^{K-k}$ for all $i \in R(k, r)$. This completes

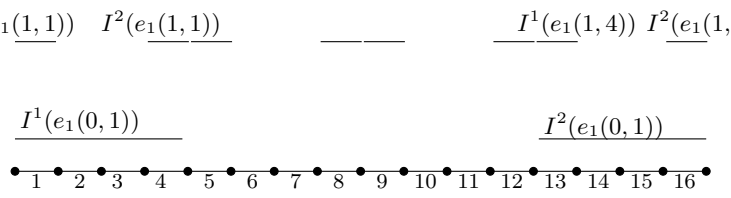


Fig. 3. The intervals corresponding to $I^l(e_1(k,r))$, where $l = 1, 2$, $k = 0, 1$ and $1 \leq r \leq n^k$, and the input graph corresponds to the instance in Figure 1

the description of how $R(k,r)$ and the corresponding sets $\mathcal{E}_i, i \in R(k,r)$ are constructed (see Figure 3 for an example). We set the values α_i equal to 1 for all $i \in R$, $p_t = 1$ for all $t \in T$ and $r_t = K \cdot \Delta$ for all $t \in T$.

In the following analysis, it will be convenient to associate a K-tuple B^t with any timeslot $t \in T$ as follows : for a fixed k, $0 \leq k \leq K-1$, let r be such that $t \in T^k_r$. As before, there are n intervals from $T^{(k+1)}$ which are contained in T^k_r – call these intervals $T^{k+1}_{s_1}, \ldots, T^{k+1}_{s_n}$ from left to right. Then the coordinate for $(k+1)$ in the vector B^t, i.e., $B^t(k+1) = l$ if $t \in T^{k+1}_{s_l}$. Observe that this vector gives the sequence of nested intervals which contain this timeslot.

Lemma 1. *If $\mathcal{I}'$ has an independent set of size A, then there is a solution to $\mathcal{I}$ of profit A^K.*

Proof. Let S be such an independent set of size A. We construct a solution for $\mathcal{I}$. For each element $e(k,r) \in R$, where $e = (v_a, v_b) \in E, 0 \leq k \leq K-1, 1 \leq r \leq r_k$, we need to pick one of the two intervals – $T^{k+1}_{s_a}$ and $T^{k+1}_{s_b}$, where s_a, s_b are the a^{th} and the b^{th} intervals of $T^{(k+1)}$ contained inside T^k_r respectively. If $v_b \notin S$, then we pick $T^{k+1}_{s_a}$, else we pick $T^{k+1}_{s_b}$. It is easy to check that in such a solution, all elements t for which $B^t = (a_1, \ldots, a_K), v_{a_i} \in S$ have $\Delta \cdot K$ distinct intervals containing it. This proves the lemma. □

Let γ denote the parameter $\frac{2^{O(\sqrt{\log \Delta})}}{\Delta}$ which appears in Theorem 4. Suppose there is no independent set of size $\gamma \cdot A$ in $\mathcal{I}'$. We first prove a technical lemma below.

Lemma 2. *Suppose we partition the edge set E into n sets — E_v for every $v \in V$, where E_v is a subset of edges incident with v. For a constant c, $0 \leq c \leq \Delta$, let $W(c) \subseteq V$ be those vertices v for which $|E_v| \geq \Delta - c$. Then, $|W(c)| \leq \gamma \cdot A \cdot (2c+1)$.*

Proof. We prove that G has an independent set of size at least $\frac{|W(c)|}{2c+1}$. In fact, this independent set will be a subset of $W(c)$. We prove this fact as follows : consider any subset $S \subseteq W(c)$, and let $G[S]$ be the subgraph of G induced by S. Then there must be a vertex in $G[S]$ of degree at most $2c$. Once we prove this fact, it is easy to check that a simple greedy algorithm will yield the desired independent set.

So fix a set $S \subseteq W(c)$. We direct each edge in $G[S]$ as follows. Suppose $e = (v, w) \in G[S]$. If $e \in E_v$, then we direct e from v to w, otherwise $(e \in E_w)$

we direct it from w to v. Since every $v \in S$ is also in $W(c)$, the indegree of any vertex is at most c. So there are at most $|S|c$ edges in $G[S]$, which implies that there must be a vertex of (undirected) degree at most $2c$ in $G[S]$. □

We are now ready to prove that $\mathcal{I}$ cannot have a solution of high profit.

Theorem 5. *Let $\mathcal{S}$ be a solution to $\mathcal{I}$ and $T^{\mathcal{S},\varepsilon}$ be the timeslots in T which are at least $(1-\varepsilon)$-satisfied. Then $|T^{\mathcal{S},\varepsilon}| \leq (10\gamma \log \Delta)^K A^K$.*

Proof. Recall that $\mathcal{E}(\mathcal{S})$ denotes the set of intervals selected by $\mathcal{S}$ – this set contains exactly one element from $\mathcal{E}_i$ for any $i \in R$. If $t \in T^{\mathcal{S},\varepsilon}$, then there are at least $\Delta \cdot K - K$ intervals in $\mathcal{E}(\mathcal{S})$ containing t. Recall that the instance $\mathcal{I}$ has exactly $\Delta \cdot K$ intervals containing t. We can associate a tuple of size K with t, which we call $Z^{\mathcal{S},t}$ as follows : fix a value k between 1 and K. Let r be such that $t \in T_r^{k-1}$. Then there are exactly Δ intervals from the sets $\mathcal{E}_{e(k,r)}, e \in E$, which contain t (and are therefore, contained in T_r^{k-1}). Among these Δ intervals, suppose $\mathcal{S}$ chooses $\Delta - c_k$ intervals. Then define $Z^{\mathcal{S},t}(k) = c_k$. Observe that $\sum_k Z^{\mathcal{S},t} \leq K$ for each t. We now define a *truncated* version $Y^{\mathcal{S},t}$ of the vector $Z^{\mathcal{S},t}$ as follows. $Y^{\mathcal{S},t}(k)$ is the nearest power of 2 greater than or equal to $Z^{\mathcal{S},t}(k)$ (unless $Z^{\mathcal{S},t}(k)$ is 0, in which case, $Y^{\mathcal{S},t}(k)$ stays as 0). We first count how many different values the vector $Y^{\mathcal{S},t}$ can take.

Claim. The number of distinct values (as a vector) that $Y^{\mathcal{S},t}$ can take is at most $(\log \Delta + 2)^K$.

Proof. Each coordinate of $Y^{\mathcal{S},t}$ is a power of 2 (or 0), and is at most 2Δ. So the number of distinct values it can take is at most $\log(2\Delta) + 1$. □

Now, we bound the number of different $t \in T^{\mathcal{S},\varepsilon}$ for which $Y^{\mathcal{S},t}$ takes a particular (vector) value.

Claim. Let $c_1, \ldots c_K$ be such that each c_i is either 0 or a power of 2, and $\sum_{k=1}^K c_k \leq 2K$. Then, the number of different $t \in T^{\mathcal{S},\varepsilon}$ for which $Y^{\mathcal{S},t} = (c_1, \ldots, c_K)$ is at most $5^K \gamma^K A^K$.

Proof. For each k, $1 \leq k \leq K$, let $r(k,t)$ be such that $t \in T^k_{r(k,t)}$. Since, $Z^{\mathcal{S},t}(k) \leq c_k$, Lemma 2 implies that for any fixed values of $T^{k'}_{r(k',t)}$ for all $k' < k$, there are at most $\gamma A \cdot (2c_k + 1)$ possibilities for $T^k_{r(k,t)}$. So the number of possible t satisfying the condition of the claim is at most $(\gamma A)^K \cdot \Pi_{k=1}^K (2c_k + 1)$. The quantity in the product (subject to $\sum_k c_k \leq 2K$) is at most 5^K. □

The above two claims now imply the theorem. □

Lemma 1 and Theorem 5, along with Theorem 4 show that it is NP-hard to get a $\left(\frac{\Delta}{2^{O(\sqrt{\Delta})}}\right)^K$-approximation for the `MaxStaff` problem even we relax the notion of satisfiability to $(1-\varepsilon)$-satisfiability for our algorithm. We now set $K = \log n$. So, the size of $\mathcal{I}$ is roughly $N = n^{\log n}$ and hence, the hardness becomes $\Omega(2^{c\sqrt{\log N}})$.

4 The Bicriteria Approximation Algorithm

In this section, we prove Theorem 2. It is not difficult to show the natural LP relaxation for this problem has a large integrality gap. We, therefore, write a stronger *configuration* LP for this problem. A configuration LP, by definition, has (perhaps exponential number of) variables which encode all possible *configurations* of a solution : these configurations should be rich enough to capture essential properties required for rounding, and yet one should be able to solve the LP in polynomial time. Getting the right balance between these two aspects turns out to be non-trivial in our setting. We motivate the ideas behind the configuration LP by looking at the following special case : requirements of all timeslots are same, i.e., $r_t = r$ for all $t \in T$. Although it requires some technical work to extend this to the general case, this special case illustrates many of the difficulties involved in coming up with a configuration LP.

4.1 The Special Case of Uniform Requirements

We consider the special case when $r_t = r$ for all timeslots. Let $\mathcal{S}$ be an optimal solution. Recall that $\mathcal{E}(S)$ denotes the set of intervals selected by $\mathcal{S}$.

Definition 1. *A* block B *is a set of* L *consecutive timeslots in* T*. We divide the timeline into* $N = \lceil \frac{T}{L} \rceil$ *blocks, namely,* $B_1, \ldots, B_N$*, in a natural manner – block* B_k *consists of timeslots* $\{(k-1)L+1, (k-1)L+2, \ldots, (k-1)L+L\}$ *(except perhaps the last block which may end early).*

To begin with, we would like to consider only a subset of blocks.

Definition 2. *We say that a set of blocks* $\mathcal{B}$ *is* independent *if no interval in* $\mathcal{E}(S)$ *intersects more than one block in* $\mathcal{B}$*. Let* $\mathcal{B}_o$ *be the set of odd numbered blocks :* $B_1, \mathcal{B}_3, \ldots$ *and* $\mathcal{B}_e$ *be the even numbered blocks : it is easy to check that* $\mathcal{B}_o$ *(or* $\mathcal{B}_e$*) is an independent set of blocks (see Figure 4).*

Since the total profit of timeslots satisfied by $\mathcal{S}$ in $\mathcal{B}_o$ or $\mathcal{B}_e$ is at least half the total profit of satisfied timeslots, we can focus (with a factor-2 loss in approximation) on an independent set of blocks – call this $\mathcal{B}$ (so, $\mathcal{B}$ is either $\mathcal{B}_e$ or $\mathcal{B}_o$). Fix a block $B \in \mathcal{B}$.

Definition 3. *A sub-block,* sB*, of* B *is a set of continuous time-slots in* B*.*

Definition 4. *Let* $\texttt{profit}(\mathcal{S}, B)$ *be the total profit of timeslots in* B *which are satisfied by* $\mathcal{S}$*. Given a sub-block* sB *of* B*, and a solution* $\mathcal{S}'$*, let* $\texttt{profit}^{\varepsilon}(\mathcal{S}', sB)$ *denote the total profit of timeslots in* sB *which are* $(1-\varepsilon)$*-satisfied by* $\mathcal{S}'$*.*

Lemma 3. *Consider a block* B *and a solution* $\mathcal{S}$*. There exists a sub-block* sB *of* B *and a solution* $\mathcal{S}'$ *consisting of a subset of intervals selected by* $\mathcal{S}$ *such that the following conditions are satisfied :*

- *The intervals in* $\mathcal{S}'$ *are mutually distinct.*
- *Every interval in* $\mathcal{S}'$ *contains the sub-block* sB*.*
- $\texttt{profit}^{\varepsilon}(\mathcal{S}', sB) \geq \varepsilon \cdot \texttt{profit}(\mathcal{S}, B)$.

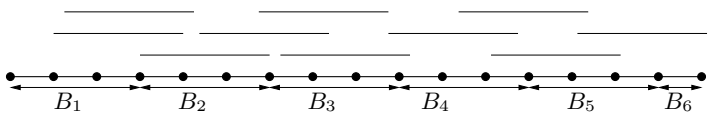


Fig. 4. The set of blocks when $L = 3$. Here, $\mathcal{B}_o = \{B_1, B_3, B_5\}, B_e = \{B_2, B_4, B_6\}$. Note that any interval picked by a solution intersects at most one block in $\mathcal{B}_o$ (or $\mathcal{B}_e$).

Proof. Let $\mathcal{S}^{\texttt{left}}$ (or $\mathcal{S}^{\texttt{right}}$) be the intervals in $\mathcal{S}$ which contain the left (or right) end-point of B – we can assume wlog that the sets $\mathcal{S}^{\texttt{left}}$ and $\mathcal{S}^{\texttt{right}}$ are disjoint (any common interval is assigned to just one of the two sets arbitrarily). Also, we can assume wlog that the intervals in $S^{\texttt{left}}$ (or $S^{\texttt{right}}$) are mutually distinct, i.e., they contain at most one interval from $\mathcal{E}_i$ for any resource $i \in R$. Indeed, we can keep the interval in $S^{\texttt{left}} \cap \mathcal{E}_i$ which goes farthest to the right and remove the rest from this set.

Let B' be the timeslots in B which are $(1-\varepsilon)$-satisfied by $\mathcal{S}$ – we denote these from left to right as as $t_1, \ldots, t_k$. We now define $\tau = \frac{1}{\varepsilon}$ disjoint sub-blocks $sB_1, sB_2, \cdots, sB_\tau$ of B. Let n_i, $1 \leq i \leq \tau$ be the smallest integer such that the number of intervals from $\mathcal{S}^{\texttt{left}}$ containing t_{n_i} is at most $(1 - i\varepsilon)r$. Define n_0 as 1. Let sB_i denote the interval $[t_{n_{i-1}}, t_{n_i})$. Consider any sub-block sB_i. Let $\mathcal{S}_1 \subseteq \mathcal{S}^{\texttt{right}}$ be the intervals which contain $t_{n_{i-1}}$ and $\mathcal{S}_2 \subseteq \mathcal{S}^{\texttt{left}}$ be those intervals which contain t_{n_i}. Let $\mathcal{K}$ denote $\mathcal{S}_1 \cup \mathcal{S}_2$. Clearly, the intervals in $\mathcal{K}$ contain sB_i, and must have at least $r(1-\varepsilon)$ distinct intervals. We are now done because we can choose the sub-block sB_i for which $\texttt{profit}^{\varepsilon}(\mathcal{S}, sB_i)$ is maximized. □

The configuration LP. We are now ready to write the configuration LP. The configurations will be defined by pairs $(B, \mathcal{A})$, where B is a block and $\mathcal{A}$ is a set of mutually distinct intervals each of which has a non-empty intersection with B. Motivated by Lemma 3, we have the following definition.

Definition 5. *For a block B, and a set of intervals $\mathcal{A}$, let $p(\mathcal{A}, B)$ denote the maximum over all sub-blocks sB of B of the quantity $\texttt{profit}^{\varepsilon}(\mathcal{A}', sB)$, where $\mathcal{A}'$ contains those intervals in $\mathcal{A}$ which contain the entire sub-block sB.*

The configuration LP has variables $y(\mathcal{A}, B)$ which is 1 when we pick all the intervals in $\mathcal{A}$, 0 otherwise.

$$\begin{aligned}
\max \quad & \sum_{B \in \mathcal{B}} \sum_{\mathcal{A}} y(\mathcal{A}, B) \cdot p(\mathcal{A}, B) \\
\sum_{\mathcal{A}: \mathcal{A} \cap \mathcal{E}_i \neq \emptyset} y(\mathcal{A}, B) &\leq \alpha_i \qquad \text{for every } i \in R \\
\sum_{\mathcal{A}} y(\mathcal{A}, B) &\leq 1 \qquad \text{for every } B \\
y(\mathcal{A}, B) &\geq 0 \qquad \text{for all } \mathcal{A}, B
\end{aligned} \tag{1}$$

It is not difficult to show that there is a polynomial time separation oracle for the dual of this LP, and so, it can be solved in polynomial time. The details are deferred to the full version.

Rounding a fractional solution. Let $y(\mathcal{A}, B)$ be an optimal solution to the configuration LP. We now show that it can be rounded to an integral solution without much loss in profit. Assume that the LP is written for $\mathcal{B} = \mathcal{B}_o$. Consider the algorithm in Figure 5. Observe that we can perform the sampling required in Step 1 because of constraint (1).

For each block $B \in \mathcal{B}_o$ do
1. Use dependent rounding to sample at most *one* of the set of intervals $\mathcal{A}$, where the probability of picking $\mathcal{A}$ is exactly $\varepsilon \cdot y(\mathcal{A}, B)$.
2. Suppose we select a set of intervals $\mathcal{A}$ in the experiment above (otherwise skip to the next step).
3. For every interval $I \in \mathcal{A}$ do
   Remove I from the set of selected intervals if we have already selected α_i intervals from the set $\mathcal{E}_i$ containing I.

Output the set of selected intervals.

Fig. 5. Algorithm `Round`

Theorem 6. *The total profit of timeslots which are* $(1 - 3\varepsilon)$*-satisfied by the solution output by the algorithm* `Round` *is at least* $\Omega(\varepsilon)$ *times the objective value of the fractional solution* $y(\mathcal{A}, B)$*.*

Proof. Note that an interval is selected only if the following two conditions hold: (i) it is picked as a part of some $\mathcal{A}$ in Step 1 of the above algorithm, and (ii) it is not dropped in Step 3 of the above algorithm. Given a set of intervals $\mathcal{A}$ picked for a specific B, the profit $p(\mathcal{A}, B)$ is accrued from some sub-block sB of B where the timeslots in sB are $(1-\varepsilon)$-satisfied by the intervals in $\mathcal{A}$. However, because of Step 3 of the algorithm, some of the intervals in $\mathcal{A}$ may get *dropped*. Each interval in $\mathcal{A}$ gets dropped (by Step 1) with a probability of at most ε. This holds even when α_i intervals are to be selected for resource i (the probability may be lower since choices of $\mathcal{A}$ according to $y(\mathcal{A}, B)$ are independent across the various blocks). Thus the expected fraction of intervals dropped is at most ε. By Markov's inequality, the probability that the number of intervals dropped is $\geq 2\varepsilon$ is at most $1/2$. Thus with probability at least $1/2$, a timeslot in sB is $(1 - \varepsilon - 2\varepsilon) = (1 - 3\varepsilon)$-satisfied by the modified set $\mathcal{A}$. Given the choice of the sets of intervals $\mathcal{A}$, it is clear from the above that (in expectation) the profit is at least $\varepsilon/2$ times the objective value of the LP. □

4.2 Extending to General Requirements

We briefly describe how we generalize the above algorithm to the case when the requirements r_t are arbitrary. As before, we can focus on a set of independent blocks $\mathcal{B}$. We first assume that the requirement of any timeslot, r_t, is a power

of $(1+\varepsilon)$. We achieve this by rounding r_t values down to the nearest power of $(1+\varepsilon)$. Since this only affects the r_t value by at most εr_t, and we are only interested in a bicriteria approximation, this assumption is valid.

Definition 6. *We say that a timeslot t is of* class c *if $r_t = (1+\varepsilon)^c$. Let $T(c)$ denote the timeslots of class c. A set T' of timeslots is said to be* well-separated *if for any two $t_1, t_2 \in T'$, where t_1 is of class c_1 and t_2 is of class c_2, either $c_1 = c_2$ or $c_1 - c_2 \geq H$. Here H is a parameter equal to $\frac{2}{\varepsilon} \cdot \ln \frac{1}{\varepsilon}$.*

We can assume that the set of timeslots are well-separated (upto a constant loss in approximation) – indeed, for a value h between 0 and $H-1$, consider only those timeslots whose class is equal to h modulo H. For any fixed value of h, these timeslots are well-separated. Since h takes only H distinct values, at least one of these values will give us at least $1/H$ fraction of the profit. So we assume that the timeslots are well-separated and let $\mathcal{C}$ denote the set of possible classes of these timeslots.

Now, given any solution $\mathcal{S}$, we would like to divide the intervals in $\mathcal{S}$ into $|\mathcal{C}|$ disjoint sets — $\mathcal{S}^c, c \in \mathcal{C}$, such that if a timeslot of class c gets satisfied by $\mathcal{S}$, then it will be at least $(1-\varepsilon)$-satisfied by $\mathcal{S}^c$. This will allow us to think of any solution as $|\mathcal{C}|$ independent solutions, one for each class $c \in \mathcal{C}$. Once we have this, then we can use the machinery developed (for uniform requirements) in the previous section for each class separately and combine the solutions. We now state this key theorem more formally. Fix a block B, and let B^c be the timeslots in B of class c. For a solution $\mathcal{S}$, define $\texttt{profit}(\mathcal{S}, \mathcal{B}^c)$ as the total profit of timeslots in $\mathcal{B}^c$ satisfied by $\mathcal{S}$. Define $\texttt{profit}^\varepsilon(\mathcal{S}, \mathcal{B}^c)$ similarly.

Theorem 7. *Fix a block B. Given a solution $\mathcal{S}$, we can partition the intervals in this solution to get solutions $\mathcal{S}^c, c \in \mathcal{C}$, such that*

$$\sum_{c \in \mathcal{C}} \texttt{profit}^\varepsilon(\mathcal{S}^c, B^c) \geq \texttt{profit}(\mathcal{S}, \cup_{c \in \mathcal{C}} B^c).$$

The proof of the theorem essentially uses the fact that there is a large gap between the requirements of timeslots belonging to two different classes. We defer the proof to the full version. Once we have this, we use the structural property in Lemma 3 for each class $c \in \mathcal{C}$ separately. This allows us to write a configuration LP. We round this LP using ideas similar to the uniform requirement case.

5 Discussion

Our algorithm can be easily extended to take care of following cases :

– Different resources may have different *proficiencies* – we say that a timeslot t is satisfied if there are a set of mutually distinct selected intervals whose total proficiency exceeds r_t. Again, we can get the same bicriteria approximation algorithm for this problem.

- Let β denote the ratio of the maximum length of an interval in $\cup_{i \in R} \mathcal{E}_i$ to the smallest length. Then we can get an $\left(\frac{\beta}{\varepsilon^3} \cdot \log \frac{1}{\varepsilon}, 1 - \varepsilon\right)$-bicriteria approximation algorithm for this problem. This allows a resource to have different shift lengths depending on when it is scheduled. The only thing that changes in the algorithm is that initially we divide the timeslots into blocks of length $L_{\min}$, where $L_{\min}$ is the smallest length of an interval. Now, instead of just looking into sets $\mathcal{B}_o$ and $\mathcal{B}_e$, we divide these blocks into β classes.
- We may stipulate that any two picked intervals from the same set $\mathcal{E}_i$ must have enough separation Δ between them; this makes sense because two different shifts of an employee should not be too close to each other. Our algorithm remains the same, except that this constraint is added in the configuration LP (and the randomized rounding procedure changes accordingly).

There are many interesting directions in which this work can be extended. In the context of contact centers, people with various *skills* may be required (say, knowledge of various languages). For a small set K of skills, an employee may have a subset of these skills. Now the requirement at time t may specify the minimum number of employees of each skill required. Modeling such problems and coming up with good approximation algorithms for them is a challenging problem. Further, instead of considering the problem of maximizing the number of timeslots satisfied, we can also consider the problem of minimizing the number of employees such that all timeslots are satisfied. However, checking feasibility of such an instance (whether all employees can be scheduled to satisfy all the timeslots) itself is NP-complete. Hence, it will be interesting to find tractable special cases of this problem. Finally, the requirements themselves are estimates. Posing these problems in a stochastic setting is also a very challenging problem.

References

[AAM07] Aksin, Z., Armony, M., Mehrotra, V.: The modern call center: A multi-disciplinary perspective on operations management research. Production and Operations Management 16(6), 665–688 (2007)

[CCPV07] Calinescu, G., Chekuri, C., Pál, M., Vondrák, J.: Maximizing a submodular set function subject to a matroid constraint (Extended abstract). In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 182–196. Springer, Heidelberg (2007)

[FHF+02] Fukunaga, A., Hamilton, E., Fama, J., Andre, D., Matan, O., Nourbakhsh, I.: Staff scheduling for inbound call centers and customer contact centers. In: AAAI, pp. 822–829 (2002)

[GKM03] Gans, N., Koole, G., Mandelbaum, A.: Telephone call centers: Tutorial, review, and research prospects. MSOM 5(2), 79–141 (2003)

[GRST10] Gupta, A., Roth, A., Schoenebeck, G., Talwar, K.: Constrained non-monotone submodular maximization: Offline and secretary algorithms. In: Saberi, A. (ed.) WINE 2010. LNCS, vol. 6484, pp. 246–257. Springer, Heidelberg (2010)

[ICWC10] Ingolfssona, A., Campelloa, F., Wub, X., Cabralc, E.: Combining integer programming and the randomization method to schedule employees. European Journal of Operations Research (2010)

[LMNS09] Lee, J., Mirrokni, V.S., Nagarajan, V., Sviridenko, M.: Non-monotone submodular maximization under matroid and knapsack constraints. In: STOC, pp. 323–332 (2009)

[Rey03] Reynolds, P.: Call Center Staffing: The Complete, Practical Guide to Workforce Management. The Call Center School Press (2003)

[ST00] Samorodnitsky, A., Trevisan, L.: A PCP characterization of NP with optimal amortized query complexity. In: STOC, pp. 191–199 (2000)

Set Covering with Ordered Replacement: Additive and Multiplicative Gaps

Friedrich Eisenbrand[1], Naonori Kakimura[2,⋆], Thomas Rothvoß[1,⋆⋆], and Laura Sanità[1,⋆⋆⋆]

[1] EPFL, Lausanne, Switzerland
[2] University of Tokyo, Japan

Abstract. We consider set covering problems where the underlying set system satisfies a particular replacement property w.r.t. a given partial order on the elements: Whenever a set is in the set system then a set stemming from it via the replacement of an element by a smaller element is also in the set system.

Many variants of Bin Packing that have appeared in the literature are such set covering problems with ordered replacement. We provide a rigorous account on the additive and multiplicative integrality gap and approximability of set covering with replacement. In particular we provide a polylogarithmic upper bound on the additive integrality gap that also yields a polynomial time additive approximation algorithm if the linear programming relaxation can be efficiently solved.

We furthermore present an extensive list of covering problems that fall into our framework and consequently have polylogarithmic additive gaps as well.

1 Introduction

Set Cover is a prominent combinatorial optimization problem that is very well understood from the viewpoint of multiplicative approximation. There exists a polynomial time factor $O(\log n)$ approximation for Set Cover [2] and a corresponding hardness result [9]. Also the (multiplicative) integrality gap of the standard linear programming relaxation for Set Cover is known to be $\Theta(\log n)$ [14].

Let $\mathcal{S}$ be a family of subsets of $[n] = \{1, \ldots, n\}$, $w : \mathcal{S} \to \mathbb{R}_+$ be a cost function and let $\chi(S) \in \{0,1\}^n$ denote characteristic vector of a set $S \in \mathcal{S}$. The Set Cover integer program

$$\min \Big\{ \sum_{S \in \mathcal{S}} w(S) x_S \mid \sum_{S \in \mathcal{S}} x_S \cdot \chi(S) \geq \mathbf{1}, x \geq \mathbf{0}, x \text{ integral} \Big\} \tag{1}$$

⋆ Supported in part by Grant-in-Aid for Scientific Research and by Global COE Program "The research and training center for new development in mathematics", MEXT, Japan.

⋆⋆ Supported by the Alexander von Humboldt Foundation within the Feodor Lynen program.

⋆⋆⋆ Supported by Swiss National Science Foundation within the project "Robust Network Design".

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 170–182, 2011.

and its linear programming relaxation is also in the focus of this paper. However, we are interested in the *additive gap* of a certain class of set covering problems. This additive gap is the *difference* between the optimum value of the integer program (1) and its linear programming relaxation. While there exists an extensive amount of literature on the (multiplicative) gap and (multiplicative) approximation algorithms, the additive gap and algorithms to construct integer solutions that are within the corresponding additive range have received less attention.

Why is it interesting to study the additive integrality gap of set covering problems? Suppose, for example that we know of a certain class of set covering problems that the additive gap is polylogarithmic, $\log n$ say. If we then, at the same time, know that the optimum solution is at least $\sqrt{n}$, then the linear programming relaxation of (1) asymptotically approaches the optimum solution of the integer program yielding a $(1+\log n/\sqrt{n})$-factor approximation algorithm if an integer solution respecting the gap can be efficiently computed.

Two prominent covering problems whose additive gap has been studied are Multi-Edge Coloring [12,16] and Bin Packing.[1] For Bin Packing, Karmarkar and Karp [13] showed that the additive gap is bounded by $O(\log^2 n)$ and they also provide a polynomial time algorithm that constructs a solution within this range. There is an extensive amount of literature on variants of Bin Packing (see e.g. [7,6,5,8,7,3,19,1]). The question whether the Set Cover linear programming relaxations of such variants also exhibit small additive gaps is in the focus of our paper.

It is easy to see that the additive gap of general Set Cover is $\Theta(n)$. For example, the Vertex Cover problem on a disjoint union of triangles exhibits this additive gap. What makes Bin Packing so special that polylogarithmic additive gaps can be shown to hold? It turns out that it is essentially the fact that in a feasible packing of a bin, we can replace any item by a smaller item and still remain feasible. In the setting of Set Cover this is reflected by the following. There is a partial order $\preceq$ of the elements that we term *replacement order*. The order is *respected* by $\mathcal{S}$ if

$$S \in \mathcal{S}, i \in S, j \notin S, j \preceq i \Rightarrow ((S \backslash \{i\}) \cup \{j\}) \in \mathcal{S}$$

We will also consider costs $w(S)$ of sets in the family $\mathcal{S}$. These costs are normalized in the sense that $w(S) \in [0,1]$ for each $S \in \mathcal{S}$. The costs *respect* the replacement order if $w(S) \geq w(S')$ whenever S' is obtained from S by replacing one element $i \in S$ with an element $j \preceq i$ and if $w(S') \leq w(S)$ for any $S' \subseteq S$. Given a family $\mathcal{S}$ and costs respecting the replacement order $\preceq$, the Set Cover With Ordered Replacement problem is to solve the integer program (1). We denote the optimum value of (1) and its relaxation by $OPT(\mathcal{S})$ and $OPT_f(\mathcal{S})$, respectively. The additive gap of $\mathcal{S}$ is thus $OPT(\mathcal{S}) - OPT_f(\mathcal{S})$.

Contributions. We provide a rigorous account on additive and multiplicative integrality gaps and approximability of Set Cover With Ordered Replacement if $\preceq$ is a total order. Our main results are as follows.

[1] Even though coined bin "packing", it is a covering problem.

- We show that the additive gap is bounded by $O(\log^3 n)$. This is achieved by the use of suitable *pseudo sizes* and grouping. The pseudo sizes are responsible for the additional $\log n$ factor compared Bin Packing. If *natural sizes* are available, our bound matches the $O(\log^2 n)$ bound for Bin Packing. The grouping technique itself is not novel albeit appears to be simpler as the one in [13].
- We provide a $\Omega(\log n)$ lower bound on the additive gap which is in contrast to Bin Packing, where such a lower bound is not known.
- We show that Set Cover With Ordered Replacement does not allow an *asymptotic polynomial time approximation scheme (APTAS)*. Also this distinguishes Set Cover With Ordered Replacement from Bin Packing.
- We show that the multiplicative gap of Set Cover With Ordered Replacement is $\Theta(\log \log n)$. Also this is in sharp contrast to Bin Packing, where the multiplicative gap is constant.
- Finally we provide a quasi-polynomial (running time $n^{O(\log n)}$) factor 2 approximation algorithm for Set Cover With Ordered Replacement.

We also bound the additive integrality gap in the case where the replacement order is not a total order. Recall that the *Dilworth number* of a partial order is the smallest number of disjoint chains that cover all elements. Let us denote the additive integrality gap as:

$$\text{gap}(n, d, k) = \max_{\mathcal{S}, w} \{OPT(\mathcal{S}) - OPT_f(\mathcal{S})\},$$

where the max ranges over all set systems over ground set $[n]$ (and proper cost function $w : \mathcal{S} \to [0, 1]$), that respect a partial order with Dilworth number d and contain sets of size at most k.[2] We show that $\text{gap}(n, d, k) = O(d^2 \log k \log^2 n)$. Our result is an *algorithmic* result in the following sense. If the linear programing relaxation of (1) can be efficiently solved, then a solution of (1) respecting the additive gap can be efficiently computed.

We furthermore demonstrate the applicability of our bounds on the additive gap by providing an extensive list of problems from the literature that can be modeled as Set Cover With Ordered Replacement.

Related work. We discuss many Bin Packing variants that are covered by our framework in Section 5. For many of these problems, there exist *asymptotic polynomial time approximation schemes (APTAS)* [10,5,3,1] or *asymptotic fully polynomial time approximation schemes (AFPTAS)* [13,7,6,8]. An AFPTAS for problem (1) is a polynomial time algorithm (in n and $1/\varepsilon$) that, given an $\varepsilon > 0$ computes a solution $APX(\mathcal{S})$ with $APX(\mathcal{S}) \leq (1 + \varepsilon)OPT(\mathcal{S}) + f(\varepsilon)$, where f is a fixed function. This function f can be even exponential in $1/\varepsilon$, see, e.g. [7,8]. While our additive gap result is incomparable with the quality achieved by an

[2] We sometimes abbreviate $\text{gap}(n, d) := \text{gap}(n, d, n)$ for systems without cardinality restrictions and $\text{gap}(n) := \text{gap}(n, 1)$ if the partial order is total.

AFPTAS it however sheds some light on how large n has to be in order for such an AFPTAS to be competitive with our result in combination with a simple constant factor approximation algorithm.

We have an additive $O(\log^3 n)$ bound for Set Cover With Ordered Replacement. If we are considering a particular family of instances with $OPT(\mathcal{S}) \geq \log^4(n)$ for each instance $\mathcal{S}$, then this yields an AFPTAS with $f(\varepsilon) = O((1/\varepsilon)^3)$. Suppose now that $OPT(\mathcal{S}) \leq \log^4(n)$ and suppose that there exists an AFPTAS with an exponential f say $f(\varepsilon) = 2^{1/\varepsilon}$. Then the dimension n has to be *doubly exponential* in $1/\varepsilon$ before the AFPTAS starts to beat the quality of a factor 2 approximation algorithm.

We would also like to mention recent related work on the additive gap of Bin Packing. The paper [4] relates a prominent conjecture of Beck from the field of *discrepancy theory* to the question whether the additive gap of Bin Packing is constant. If Beck's conjecture holds true, then the Bin Packing gap is constant for 3-partitioning instances. While essentially all results on integrality gaps for Bin Packing variants in the literature use the sparse support of basic solutions, [17] provides bounds based on probabilistic techniques. The work in [17] provides better additive gaps for example in case of Bin Packing With Rejection.

2 Bounding the Additive Gap

In this section we provide upper and lower bounds for the additive integrality gap of Set Cover With Ordered Replacement. The upper bound in case of a total order is $O(\log^3 n)$ while the lower bound is of order $\Omega(\log n)$. This result shows that in order to have a polylogarithmic additive integrality gap, it is sufficient to have a total ordering on the elements, like for classical Bin Packing.

2.1 The Upper Bound

We first deal with an upper bound on the additive integrality gap. We recall that d is the Dilworth number of the partial order, n denotes the number of elements and k is an upper bound on the cardinality of the sets in the family $\mathcal{S}$. We show the following general theorem.

Theorem 1. *One has* $gap(n, d, k) = O(d^2 \log k \log^2 n)$.

As in [13], we will construct an integer solution to (1) from a fractional one with the claimed additive bound, by doing a sequence of iterations. At each iteration, we will cover part of our elements by rounding down an optimal fractional solution of a proper updated linear program, modify the residual instance, and re-optimize.

More precisely, we will consider the following (more general) linear program

$$\min \Big\{ \sum_{S \in \mathcal{S}} w(S) x_S \mid \sum_{S \in \mathcal{S}} x_S \cdot \chi(S) \geq b, x \geq \mathbf{0} \Big\}, \tag{2}$$

where $b \in \mathbb{N}_0^n$ is a non-negative vector. The number b_i denotes the *multiplicity* of the item i, i.e., how many times it needs to be covered. The reason of having multiplicity is that during our iterations, we will reduce the number of constraints of the above linear program at the expense of increasing multiplicity for a subset of the items. Of course, when multiplicity comes into play, we allow $S \in \mathcal{S}$ to a be multiset, since e.g. if a set $S' = \{i, j, h\} \in \mathcal{S}$ and $i \preceq j \preceq h$, the replacement order implies $S = \{i, i, i\}$ to be in $\mathcal{S}$ as well.

Let us denote the optimum of this linear program with multiplicity and corresponding integer program by $OPT_f(\mathcal{S}, b)$ and $OPT(\mathcal{S}, b)$ respectively. As in [13] we will consider an optimal vertex solution x^* of the linear program (2) and extract the partial covering $\lfloor x^*_S \rfloor$, $S \in \mathcal{S}$. This partial covering will leave some elements of the multiset uncovered. This multiset of uncovered elements defines a *residual instance*, encoded by $b' = \sum_{S \in \mathcal{S}} \{x^*_S\} \cdot \chi(S)$ where $\{x^*_S\} = x^*_S - \lfloor x^*_S \rfloor$ denotes the fractional part of x^*. The following relation holds even for arbitrary set covering problems:

$$OPT(\mathcal{S}, b) - OPT_f(\mathcal{S}, b) \leq OPT(\mathcal{S}, b') - OPT_f(\mathcal{S}, b'). \tag{3}$$

The key point of considering a residual instance, is that we will modify it to reduce the number of constraints in the next iteration by using a grouping technique similar to the one given for Bin Packing in [13]. However, the grouping in [13] crucially relies on the *size of an element* which is part of a Bin Packing instance, and that is instead missing in our abstract setting. In order to still apply grouping techniques, we will define *pseudo sizes* $s_i \in]0, 1]$ for each element i in our ground set below. These pseudo sizes satisfy the following properties:

(i) $s_i \geq s_j$ if $i \succeq j$;
(ii) We can cover all the elements of any (not necessarily maximal) chain C at cost $O(\sum_{i \in C} s_i) + O(\log \frac{1}{s_{\min}})$, where[3] $s_{\min} := \min_{i \in [n]} s_i$.

Notice that the usual size of an element in a Bin Packing instance is also a pseudo size[4]. Given a vector of pseudo sizes s, we can define the *total size* of the elements by $\text{size}(b) = b^T s$. Suppose now that the largest total pseudo size of a set in $\mathcal{S}$ is $\alpha = \max\{\sum_{i \in S} s_i \mid S \in \mathcal{S}\}$. An important observation is that the total size of the residual instance is bounded by

$$\text{size}(b') = {b'}^T s = \sum_{S \in \mathcal{S}} \{x^*(S)\} \chi(S)^T s \leq \text{support}(b) \cdot \alpha, \tag{4}$$

where $\text{support}(b)$ denotes the number of nonzero entries in b.

We are now ready to state the following Lemma, which generalizes the result of Karmakar and Karp [13].

Lemma 1. *Let $\mathcal{S}$ be a set system on $[n]$ with cost function $w : \mathcal{S} \to [0, 1]$, $\preceq$ be a partial order respected by $\mathcal{S}$ of Dilworth number d and let $b \in \mathbb{N}_0^n$. Still,*

[3] For notational convinience, we always assume that $s_{\min} \leq 1/2$ so that $\log \frac{1}{s_{\min}} \geq 1$.
[4] Here one can cover a subset of elements even with $2 \sum_{i \in C} s_i + 1$ bins alone and the minimal size does not need to be considered.

let s be a vector of pseudo sizes which satisfies properties (i) and (ii). Then $OPT(\mathcal{S}, b) - OPT_f(\mathcal{S}, b) = O(\alpha \log(1/s_{\min}) \cdot d \log n)$, where $\alpha = \max\{\sum_{i \in S} s_i \mid S \in \mathcal{S}\}$.

Proof. Initially, we partition the elements into chains $C_1, \ldots, C_d$ w.r.t. the order $\succeq$. The bound follows from iterating the following procedure. First, we replace b by b' which encodes its residual instance. According to (3), this does not decrease the additive integrality gap. We then apply the following grouping procedure to each chain C_μ with $\mu = 1, \ldots, d$. The chain C_μ is partitioned into *classes*:

$$U_\ell^\mu = \left\{ i \in C_\mu \mid \left(\frac{1}{2}\right)^{\ell+1} < s_i \leq \left(\frac{1}{2}\right)^{\ell} \right\} \text{ for } \ell = 0, \ldots, \lfloor \log(1/s_{\min}) \rfloor.$$

For each such class U_ℓ^μ, build groups of $4 \cdot 2^\ell \alpha$ consecutive elements (the elements are always counted with multiplicity), starting from the largest element (the last group could contain less elements), and discard the first and the last group. In this way, we discard at most $8 \cdot 2^\ell \cdot \alpha$ elements in the class U_ℓ^μ. Those elements have total size at most $8 \cdot \alpha$, hence the total size of discarded elements in chain C_μ is bounded by $8 \cdot \alpha \cdot (\log(1/s_{\min}) + 1)$. By (ii) we can cover them at cost $O(\alpha \cdot \log(1/s_{\min}))$. This amounts to a cost of $O(d \cdot \alpha \cdot \log(1/s_{\min}))$ to cover all discarded elements of all chains.

Then, we "round-up" the elements in each group to the largest element in this group. In other words, for each group we now consider to have one item type (the largest one, according to the chain) with multiplicity $4 \cdot 2^\ell \alpha$. This way, we obtain a new "rounded" instance that is represented by a vector $b'' \in \mathbb{N}_0^n$. Since the discarded groups compensate the round-up operation within each group, one has $OPT_f(\mathcal{S}, b'') \leq OPT_f(\mathcal{S}, b')$. Also, $OPT(\mathcal{S}, b') \leq OPT(\mathcal{S}, b'') + O(d \cdot \alpha \cdot \log(1/s_{\min}))$ and thus

$$OPT(\mathcal{S}, b') - OPT_f(\mathcal{S}, b') \leq OPT(\mathcal{S}, b'') - OPT_f(\mathcal{S}, b'') + O(d \cdot \alpha \cdot \log(1/s_{\min})).$$

We will next show that $\text{support}(b'') \leq \text{support}(b)/2$. The assertion then follows, since the support of the rounded instance (and hence the corresponding additive integrality gap) will be 1 after $O(\log n)$ iterations of the above described procedure.

The support of b'' is the number of non-discarded groups. Each non-discarded group U_ℓ^μ contains a number of elements equal to $4 \cdot 2^\ell \alpha$, each of size at least $\frac{1}{2^{\ell+1}}$. Then the total size of the group is at least $2 \cdot \alpha$. Thus $2 \cdot \alpha \cdot \text{support}(b'') \leq \text{size}(b')$. But since b' encodes a residual instance, one has $\text{size}(b') \leq \text{support}(b) \cdot \alpha$ from (4). That is, $\text{support}(b'') \leq \text{support}(b)/2$ follows. □

Notice that the $O(\log^2 n)$ upper bound of Karmarkar and Karp [13] for Bin Packing also follows from this lemma by considering the original sizes given in the instances as pseudo sizes, and setting d, α and all the initial b_i's to one. Items of size less than $1/n$ can be removed and distributed on top of the solution later. If one needs an additional bin, the gap is even bounded by a constant.

We now show how to define good pseudo sizes of the items for *any* given set system $\mathcal{S}$. Let

$$\text{size}(i) := \min\left\{ \frac{w(S)}{|S|} \mid S \in \mathcal{S} \text{ contains only elements } j \succeq i \right\}$$

Note that $\text{size}(i) \geq \text{size}(j)$ holds if $i \succeq j$, that is, property (i) of Lemma 1 holds. The next Lemma shows that also (ii) holds as well.

Lemma 2. *Let $\mathcal{S}$ be a set system with replacement property. We can cover all the elements of a (not necessarily maximal) chain C at cost at most $2\sum_{i\in C} \text{size}(i) + O(\log \frac{1}{s_{\min}})$.*

Proof. Again let $U_\ell = \{i \in C \mid (\frac{1}{2})^{\ell+1} < \text{size}(i) \leq (\frac{1}{2})^{\ell}\}$ be the ℓth size class of chain C. We construct a solution for each size class separately. For class ℓ, consider iteratively the largest uncovered element i and let $S(i)$ be a set, defining the quantity $\text{size}(i)$, i.e. $S(i)$ contains only elements that are at least as large as i and $\frac{w(S(i))}{|S(i)|} \leq \text{size}(i)$. By the replacement property, this set $S(i)$ can be used to cover the largest $|S(i)|$ many uncovered elements.

Let $S_1, \ldots, S_p$ be the sets selected in this procedure. Note that all the S_j, but possibly S_p, cover exactly $|S_j|$ elements. Then, since $\frac{w(S_j)}{|S_j|} \leq 2\text{size}(i)$ for every $i \in S_j$, we have

$$\sum_{j=1}^{p} w(S_j) = \sum_{j=1}^{p-1} \sum_{i\in S_j} \frac{w(S_j)}{|S_j|} + w(S_p) \leq \sum_{i\in U_\ell} 2 \cdot \text{size}(i) + 1.$$

Summing over the $(\log \frac{1}{s_{\min}} + 1)$ many size classes then gives the claim. □

We are ready to prove Theorem 1.

Proof (Proof of Theorem 1). Let $\mathcal{S}$ be a set system on $[n]$, and $\preceq$ be a partial order respected by $\mathcal{S}$. We may assume that $\preceq$ consists of d incomparable chains $C_1, \ldots, C_d$. We define $s_i = \text{size}(i)$ for any element i.

We first claim that we can discard all items i with $s_i < \frac{1}{n^2}$: in fact, by the definition of the pseudo sizes any such element is contained in a set of cost at most $\frac{1}{n}$, hence all such tiny elements can be covered at a total cost of 1.

Let $S \in \mathcal{S}$ and consider an element i which is the jth largest in $S \cap C_\mu$. By the definition of the pseudo sizes, $\text{size}(i) \leq \frac{w(S)}{j}$. Thus

$$\sum_{i\in S} \text{size}(i) \leq d \sum_{j=1}^{k} \frac{w(S)}{j} \leq 2d \log k.$$

Therefore, it follows from Lemma 1 (by setting $b = \mathbf{1}$) that $\text{gap}(n, d, k) = O(d^2 \log k \log^2 n)$, since $\alpha = 2d \log k$ and $s_{\min} \geq 1/n^2$. □

We want to remark that the above result is constructive in the sense that once a fractional solution x and the partial order are given, a feasible integer solution matching the bound of Theorem 1 can be computed in polynomial time.

If all costs are one, i.e., $w(S) = 1$ for each $S \in \mathcal{S}$, then we have size$(i) \geq \frac{1}{k}$. Thus we have the following corollary.

Corollary 1. *If all costs of feasible sets are one, one has* $gap(n, d, k) = O(d^2 \log^2 k \log n)$.

When $d = 1$, this theorem says $\text{gap}(n, 1, k) = O(\log^2 k \log n)$, which is better than the result of [4]: $\text{gap}(n, 1, k) = O(k \log n)$.

2.2 The Lower Bound

In this section, we give a lower bound for the additive integrality gap of set systems with replacement property. For simplicity, we first assume that $\preceq$ consists of only one chain.

Lemma 3. *One has* $gap(n) \geq \Omega(\log n)$.

Proof. Let m be a parameter, that we determine later. Define a unit cost set system $\mathcal{S}$ as follows. For any $\ell \in \{1, \ldots, m\}$, introduce $3 \cdot 100^\ell$ many *ℓ-level elements* U_ℓ. Hence $U := \bigcup_{\ell=1}^m U_\ell$ with $|U_\ell| = 3 \cdot 100^\ell$ is the set of all elements. We choose $U_1 \succeq U_2 \succeq \ldots \succeq U_m$[5] and an arbitrary order within every U_ℓ. Any set S of at most $2 \cdot 100^\ell$ many ℓ-level elements forms an ℓ-level set, e.g. $\mathcal{S} = \bigcup_{\ell=1}^m \left\{ S \subseteq U_\ell \cup \ldots \cup U_m \mid |S| \leq 2 \cdot 100^\ell \right\}$. By taking 3 ℓ-level sets to the fractional extend of $\frac{1}{2}$, we can cover all ℓ-level elements, hence $OPT_f(\mathcal{S}) \leq \frac{3}{2} \cdot m$. It remains to lower bound $OPT(\mathcal{S})$. Let n_ℓ be the number of ℓ-level sets in any integer solution. To cover all level ℓ-level elements we must either have $n_\ell \geq 2$, or $\sum_{j<\ell} n_j \cdot 2 \cdot 100^j \geq 100^\ell$, i.e. $4 \sum_{j<\ell} n_j \cdot 100^{j-\ell} \geq 2$. In any case, the sum of the left hand sides must be at least 2: $n_\ell + 4 \sum_{j<\ell} n_j \cdot 100^{j-\ell} \geq 2$. We add up this inequality for $\ell = 1, \ldots, m$ and obtain $(*)$

$$\sum_{\ell=1}^{m} n_\ell \cdot \underbrace{\left(1 + 4 \sum_{i \geq 1} \frac{1}{100^i}\right)}_{=\frac{103}{99}} \geq \sum_{\ell=1}^{m} \left(n_\ell + 4 \sum_{j<\ell} n_j \cdot 100^{j-\ell}\right) \overset{(*)}{\geq} 2m$$

This gives $OPT(\mathcal{S}) - OPT_f(\mathcal{S}) \geq \sum_{\ell=1}^m n_\ell - \frac{3}{2}m \geq 2\frac{99}{103}m - \frac{3}{2}m > 0.4 \cdot m$. The number of elements in the instance is $n = \sum_{\ell=1}^m 3 \cdot 100^\ell$, hence $m = \Omega(\log n)$. □

More generally, the following holds:

Theorem 2. $gap(n, d) \geq \Omega(d \cdot \log(n/d))$.

Proof. Apply the construction from Lemma 3 to obtain families $\mathcal{S}_1, \ldots, \mathcal{S}_d$, each on a disjoint set of n/d elements. Then the union $\bigcup_{j=1}^d \mathcal{S}_j$ has Dilworth number d and the claimed additive gap. □

[5] We abbreviate $U \succeq U' \Leftrightarrow \forall i \in U, j \in U' : i \succeq j$.

3 Approximation and Intractability

In Section 5, we mention many variants of Bin Packing that admit an APTAS (or even an AFPTAS) and fall into our general framework of Set Cover With Ordered Replacement. It is thus natural to ask whether the Set Cover With Ordered Replacement itself has an APTAS. This cannot be expected for arbitrary partial orders. In this section we show that an APTAS does not exist, even if the replacement order is a total order. On the positive side however, we show that there exists a quasi-polynomial time factor 2 approximation algorithm in this case. This is obtained by first rounding the instance to $O(\log n)$ different item types and then solving the rounded instance exactly by dynamic programming. However, due to lack of space, we postpone the proof to the full version of this paper.

From now on, we restrict our view exclusively on set systems, respecting a total order $\preceq$. To define such a set system with unit-costs, it will be convenient to consider just the set of *generators*. These are the sets that are maximal with respect to the order. More formally, if there is an injective map $\varphi : S \to S'$ with $i \preceq \varphi(i)$ for all $i \in S$ (i.e. we can obtain a set S from S' by applying the replacement rule), then we say that S' *dominates* S. Hence a set family $\mathcal{S}$ is called the set of *generators* for the set system

$$g(\mathcal{S}) = \{S \subseteq [n] \mid \exists S' \in \mathcal{S} : S' \text{ dominates } S\}$$

Hence, if $\mathcal{S}$ is an arbitrary set family, by definition $g(\mathcal{S})$ respects the replacement rule. For a proof of the next proposition we refer to the full version of this paper.

Proposition 1. *Let $\mathcal{S} \subseteq 2^{[n]}$ be a family of sets and $\succeq$ the total order with $1 \succeq \ldots \succeq n$.*

i) If $\mathcal{S}' \subseteq g(\mathcal{S})$ is a feasible solution (i.e. $\bigcup_{S \in \mathcal{S}'} S = [n]$) then $\sum_{S \in \mathcal{S}'} |S \cap \{1, \ldots, i\}| \geq i$ for $i = 1, \ldots, n$.

ii) If $\mathcal{S}' \subseteq \mathcal{S}$ are generators with $\sum_{S \in \mathcal{S}'} |S \cap \{1, \ldots, i\}| \geq i \; \forall i \in [n]$, then sets $\mathcal{S}'$ can be replaced by dominated ones which form a feasible solution of the same cardinality.

3.1 Ruling Out an APTAS

We will now see that unless $\mathbf{P} = \mathbf{NP}$, there is no APTAS for a generic problem defined on a set system that respects a total order.

Theorem 3. *For every $\varepsilon > 0$ and any $C > 0$ there is a generic problem with unit-cost sets respecting a total order for which it is* **NP***-hard to find a solution of cost $(\frac{3}{2} - \varepsilon)OPT + C$.*

Proof. We will prove the theorem by constructing a set system such that for any fixed integer $k > 0$, it is **NP**-hard to distinguish whether an optimum solution consists of at most $2k$ or at least $3k$ sets. Choosing $k := k(\varepsilon, C)$ large enough then gives the claim.

To establish this hardness result, we will use a reduction from the **NP**-hard PARTITION [11] problem. An instance $\mathcal{I}$ of PARTITION, is given by a set of n items with sizes $a_1 \geq a_2 \geq \ldots \geq a_n$, and asks for a partition of the items into two sets A_1 and A_2, such that $\sum_{j\in A_1} a_j = \sum_{j\in A_2} a_j =: A$. Given such an instance, we create groups $L_1, \ldots, L_k$, where the group L_p contains n^{2p} copies of item i, i.e. $L_p = \{v^j_{p,i} \mid i = 1, \ldots, n; j = 1, \ldots, n^{2p}\}$. Note that the total number of elements is $N := n^{O(k)}$. We define a total order with $L_1 \succeq \ldots \succeq L_k$ and $v^j_{p,i} \succeq v^{j'}_{p,i'}$ whenever $i < i'$ (and $v^1_{p,i} \succeq \ldots \succeq v^{n^{2p}}_{p,i}$ for the sake of completeness). Let $S(I,p) := \{v^j_{p,i} \mid i \in I;\ j = 1, \ldots, n^{2p}\}$ be the set induced by $I \subseteq [n]$ in group L_p. We define generators

$$\mathcal{S} = \left\{ S(I,p) \mid \forall p = 1, \ldots, k;\ \forall I \subseteq [n] : \sum_{i\in I} a_i \leq A \right\}$$

Completeness: $\mathcal{I} \in$ PARTITION $\Rightarrow OPT(g(\mathcal{S})) \leq 2k$. Let $I \subseteq [n]$ with $\sum_{i\in I} a_i = A$ be a PARTITION solution. Then the $2k$ sets of the form $S([n]\backslash I, p), S(I,p)$ for $p = 1, \ldots, k$ cover all N elements.
Soundness: $\mathcal{I} \notin$ PARTITION $\Rightarrow OPT(g(\mathcal{S})) \geq 3k$. We may assume that $3k < n$. Now suppose for the sake of contradiction, there is no PARTITION solution, but $\mathcal{S}' \subseteq \mathcal{S}$ is a family of less than $3k$ generating sets, dominating a feasible solution, satisfying the second condition in Prop. 1. Then there must be a group L_p such that $\mathcal{S}'$ contains generators $S(I_1,p), \ldots, S(I_m,p)$ with $m \leq 2$. Then from Prop. 1 we obtain that

$$i \cdot n^{2p} \overset{\text{Prop. 1}}{\leq} \sum_{S\in\mathcal{S}'} |S \cap (L_1 \cup \ldots \cup L_{p-1} \cup S(\{1,\ldots,i\},p))| \leq 3k \cdot n \cdot n^{2p-2} + \sum_{\ell=1}^{m} n^{2p} |I_\ell \cap \{1,\ldots,i\}|$$

Hence: $\sum_{\ell=1}^{m} |I_\ell \cap \{1,\ldots,i\}| \geq \left\lceil i - \frac{3k}{n} \right\rceil = i$, since the left hand side is integral and $3k < n$. Since $\sum_{\ell=1}^{m} |I_\ell \cap \{1,\ldots,i\}| \geq i$ for all $i \in [n]$, we conclude by applying again Prop. 1, that elements in $I_1, \ldots, I_m$ can be replaced by smaller ones and obtain $I'_1, \ldots, I'_m$ such that still $\sum_{i\in I'_\ell} a_i \leq A$ but $\bigcup_{\ell=1}^{m} I'_\ell = [n]$. Since $m \leq 2$, this is a contradiction. □

4 Multiplicative Integrality Gaps

Next, we show a $\Theta(\log\log n)$ multiplicative gap for the case of a total order.

Lemma 4. *Let $\mathcal{S}$ any set family on n elements respecting a total order and let $w : \mathcal{S} \to [0,1]$ be a cost function. Then $OPT(\mathcal{S}) \leq O(\log\log n) \cdot OPT_f(\mathcal{S})$.*

Proof. Consider consecutive groups $L_0, \ldots, L_k$ such that $|L_i| = 2^i$ (group L_k might have fewer elements), $k \leq \log n$ and all elements in L_i are larger than those in L_{i+1}. Let $x \in [0,1]^{\mathcal{S}}$ be a fractional solution. We buy set S independently with probability $\lambda \cdot x_S$ where $\lambda := \max\{8 \cdot \log(4 + 4\log n), 4\}$ (in fact we may assume that $\lambda \cdot x_S \leq 1$, otherwise we buy $\lfloor \lambda x_S \rfloor$ sets deterministically and then

another set with probability $\lambda x_S - \lfloor \lambda x_S \rfloor$). Let X_S be the indicator variable, telling whether we bought S or not. Define $E_i := \{\sum_S X_S \cdot |S \cap L_i| < 2 \cdot |L_i|\}$ as the event that we bought less than two times enough sets for the ith group. Recall the following Chernov bound (see[6] e.g. [15]):

Let $Y_1, \ldots, Y_m$ be independent random variables with $Y_i \in [0,1]$, $Y = \sum_{i=1}^m Y_i$ and $0 < \delta < 1$. Then $\Pr[Y \leq (1-\delta)E[Y]] \leq e^{-E[Y]\delta^2/2}$.

Applying Chernov bound with $\delta := 1/2$, $Y_S := X_S \frac{|S \cap L_i|}{|L_i|}$ and $E[Y] = \lambda$, we obtain

$$\Pr[E_i] = \Pr\Big[\sum_{S \in \mathcal{S}} X_S \cdot |S \cap L_i| < 2 \cdot |L_i|\Big] \leq \Pr\Big[\sum_{S \in \mathcal{S}} Y_S < (1-\delta)\lambda\Big] \leq e^{-\lambda/8} \leq \frac{1}{4(1+\log n)}.$$

By the union bound, $\Pr[E_0 \cup \ldots \cup E_k] \leq \frac{1}{4}$. Furthermore $\Pr[\sum_{S \in \mathcal{S}} X_S w(S) > 4\lambda \cdot OPT_f] \leq \frac{1}{4}$ by Markov's inequality. Overall, with probability at least $1/2$, we obtain an integer solution $\mathcal{S}' = \{S \in \mathcal{S} \mid X_S = 1\}$ of cost at most $O(\log\log n) \cdot OPT_f$ that reserves at least $2|L_i|$ slots for elements in L_i. Those slots are enough to cover all elements in L_{i+1}. □

Note that the result in Lemma 4 provides a randomized polynomial time algorithm, provided that a near-optimal fractional solution x can be obtained.

Lemma 5. *There exists a set system $\mathcal{S}' \subseteq 2^{[n]}$ with unit cost sets and respecting a total order such that $OPT(\mathcal{S}') \geq \Omega(\log\log n) \cdot OPT_f(\mathcal{S}')$.*

Proof. Let $k \in \mathbb{N}$ be a parameter. To construct our instance, we use as starting point a Set Cover instance defined by a set system $\mathcal{C}$ with $2^k - 1$ sets $C_1, \ldots, C_{2^k-1}$ and $2^k - 1$ elements $U = \{1, \ldots, 2^k - 1\}$ such that one needs at least k sets to cover all elements, while $OPT_f(\mathcal{C}) \leq 2$ (see Example 13.4 in the book of Vazirani [18]).

For every element $i \in U$, create *groups* L_i with $|L_i| = (2k)^i$. For any C_j in the original set system, define a set $S_j := (\bigcup_{i \in C_j} L_i)$ with unit cost, and let $\mathcal{S} = \{S_1, \ldots, S_{2^k-1}\}$. In other words, we take a standard Set Cover instance and replace the ith element by $(2k)^i$ elements. We define a total order such that all items in L_i are larger than those in L_{i+1} (and any order within the groups). The claim is that the set system $\mathcal{S}' := g(\mathcal{S})$, which is generated by the sets $\mathcal{S}$, has a covering integrality gap of at least $k/2$. First note that still $OPT_f(\mathcal{S}') \leq 2$. Now suppose for contradiction that there are generators (after reindexing) $S_1, \ldots, S_m \subseteq \mathcal{S}$ with $m < k$ that satisfy the condition in Prop. 1. Since $m < k$, there must be an index i such that $i \notin (C_1 \cup \ldots \cup C_m)$. Then

$$(2k)^i \leq \sum_{\ell=1}^{i} |L_\ell| \overset{\text{Prop. 1}}{\leq} \sum_{j=1}^{m} |S_j \cap (L_1 \cup \ldots \cup L_i)| = \sum_{j=1}^{m} \sum_{\ell \in C_j, \ell \leq i} (2k)^\ell \overset{i \notin C_j}{\leq} m \cdot 2(2k)^{i-1} = \frac{m}{k}(2k)^i$$

Rearranging yields that $m \geq k$. Note that the number of elements in the system $\mathcal{S}'$ is $n = \sum_{i=1}^{2^k-1} (2k)^i \leq 2 \cdot (2k)^{2^k}$, hence $k = \Omega(\log\log n)$. □

[6] To be precise, the claim in [15] is for 0/1 distributed random variables, but the same proof goes through if $0 \leq Y_i \leq 1$.

5 Applications

We now demonstrate the versatility of our framework by establishing small additive integrality gaps for several Bin Packing variants from the literature, listed in the following. We refer to the full version of this paper for the proofs of the bounds provided below.

Cardinality Bin Packing. Here we are given a Bin Packing instance $s_1, \ldots, s_n$ with an additional parameter $k \in \mathbb{N}$. A subset of items S can be assigned to a bin only if $\sum_{i \in S} s_i \leq 1$ and $|S| \leq k$. There is an AFPTAS due to Epstein & Levin [7]. With our technique, we obtain: $OPT(\mathcal{S}) - OPT_f(\mathcal{S}) = O(\log k \log n)$.

Open End Bin Packing. We are here given a Bin Packing instance $s_1, \ldots, s_n$, but a set $S \subseteq [n]$ is feasible if $\sum_{i \in S \setminus \{j\}} s_i \leq 1$ holds for every $j \in S$. There is an AFPTAS for this variant by Epstein & Levin [6]. We can prove that: $OPT(\mathcal{S}) - OPT_f(\mathcal{S}) = O(\log^2 n)$.

Bin Packing with General Cost Structure. We are given item sizes $s_1, \ldots, s_n$, and a cost function $f : \{0, \ldots, n\} \to [0, 1]$, which is a monotonically nondecreasing concave function with $f(0) = 0$. The goal is to find sets $S_1, \ldots, S_p$ to cover the items such that $\sum_{i \in S_j} s_i \leq 1$ for every $j = 1, \ldots, p$ and the total cost $\sum_{i=1}^{p} f(|S_i|)$ is minimized. This problem admits an AFPTAS [8]. We prove: $OPT(\mathcal{S}) - OPT_f(\mathcal{S}) = O(\log^2 n)$.

Generalized Cost Variable Sized Bin Packing. We are given item sizes $s_1, \ldots, s_n$, and bin types $B_1, \ldots, B_k$ each one with a different capacity $a_1, \ldots, a_k$ and a different cost $c_1, \ldots, c_k \in [0, 1]$. The goal is to select a minimum cost subset of bins to pack the items, where a subset of items can be packed into a bin B_i if the sum of their sizes does not exceed the bin capacity a_i. See [5] for an APTAS. Our result is: $OPT(\mathcal{S}) - OPT_f(\mathcal{S}) = O(\log^3 n)$.

Bin Packing with Rejection. We are given a Bin Packing instance $s_1, \ldots, s_n$, but additionally any item i has a rejection cost c_i. An item can either be packed, or discarded at cost c_i. See [7] for an AFPTAS. We have: $OPT(\mathcal{S}) - OPT_f(\mathcal{S}) = O(\sqrt{n} \log^{3/2} n)$.

Train Delivery. We are given n items, each having a size s_i and position $p_i \in [0, 1]$. The goal is to transport all items to a depot, located at 0, using a unit capacity train and minimizing the total tour length. The authors in [3] give an APTAS. We can prove that: $OPT(\mathcal{S}) - OPT_f(\mathcal{S}) = O(\sqrt{n} \log^{3/2} n)$.

m-dimensional Vector Packing. Let V be a set of vectors $v_1, \ldots, v_n \in [0, 1]^m$. The goal is to partition V into bins $B_1, \ldots, B_k$, such that k is minimized and $\sum_{i \in B_j} v_i \leq \mathbf{1}$. We say that $v_i \preceq v_j$, if v_j is componentwise not smaller than v_i. The Dilworth number of V is then the smallest d such that V can be partitioned into $V^1, \ldots, V^d$, such that: $\forall v_i, v_j \in V^h$, either $v_i \preceq v_j$ or $v_j \preceq v_i$. If there is no bound on d, there is no APTAS possible already in 2-dimensions [19]. Differently, for constant d there is an APTAS given in [1]. Our result is: $OPT(\mathcal{S}) - OPT_f(\mathcal{S}) = O(d^2 \log^3 n)$.

References

1. Caprara, A., Kellerer, H., Pferschy, U.: Approximation schemes for ordered vector packing problems. Naval Research Logistics 50, 58–69 (2003)
2. Chvátal, V.: A greedy heuristic for the set-covering problem. Mathematics of Operations Research 4(3), 233–235 (1979)
3. Das, A., Mathieu, C., Mozes, S.: The train delivery problem- vehicle routing meets bin packing (2010)
4. Eisenbrand, F., Pálvölgyi, D., Rothvoß, T.: Bin packing via discrepancy of permutations. In: Symposium on Discrete Algorithms, SODA 2011. SIAM, Philadelphia (2011)
5. Epstein, L., Levin, A.: An APTAS for generalized cost variable-sized bin packing. SIAM Journal on Computing, 38(1) (2008)
6. Epstein, L., Levin, A.: Asymptotic fully polynomial approximation schemes for variants of open-end bin packing. Inf. Process. Lett. 109(1), 32–37 (2008)
7. Epstein, L., Levin, A.: AFPTAS results for common variants of bin packing: A new method to handle the small items. SIAM Journal on Optimization (2010) (to appear)
8. Epstein, L., Levin, A.: Bin packing with general cost structures. Mathematical Programming (2010) (to appear)
9. Feige, U.: A threshold of ln n for approximating set cover. Journal of the ACM 45(4), 634–652 (1998)
10. Fernandez de la Vega, W., Lueker, G.S.: Bin packing can be solved within $1+\varepsilon$ in linear time. Combinatorica 1(4), 349–355 (1981)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York (1979)
12. Kahn, J.: Asymptotics of the chromatic index for multigraphs. Journal of Combinatorial Theory. Series B 68(2), 233–254 (1996)
13. Karmarkar, N., Karp, R.M.: An efficient approximation scheme for the one-dimensional bin-packing problem. In: 23rd annual symposium on foundations of computer science (Chicago, Ill., 1982), pp. 312–320. IEEE, New York (1982)
14. Lovász, L.: On the ratio of optimal integral and fractional covers. Discrete Mathematics 13(4), 383–390 (1975)
15. Mitzenmacher, M., Upfal, E.: Probability and computing. Cambridge University Press, Cambridge (2005); Randomized algorithms and probabilistic analysis
16. Plantholt, M.: A sublinear bound on the chromatic index of multigraphs. Discrete Mathematics 202(1-3), 201–213 (1999)
17. Rothvoß, T.: The entropy rounding method in approximation algorithms (2010) (submitted)
18. Vazirani, V.V.: Approximation Algorithms. Springer, Heidelberg (2001)
19. Woeginger, G.J.: There is no asymptotic PTAS for two-dimensional vector packing. Information Processing Letters 64(6), 293–297 (1997)

Backdoor Branching

Matteo Fischetti and Michele Monaci

DEI, University of Padova, via Gradenigo 6/A, 35131 Padova, Italy
{matteo.fischetti,michele.monaci}@unipd.it

Abstract. Which is the minimum number of variables that need branching for a given MIP instance? Can this information be effective in producing compact branching trees, hence improving the performance of a state-of-the-art solver? In this paper we present a restart exact MIP solution scheme where a set covering model is used to find a small set of variables (a "backdoor", in the terminology of [8]) to be used as first-choice variables for branching. In a preliminary "sampling" phase, our method quickly collects a number of relevant low-cost fractional solutions that qualify as obstacles for LP bound improvement. Then a set covering model is solved to detect a small subset of variables (the backdoor) that "cover the fractionality" of the collected fractional solutions. These backdoor variables are put in a priority branching list, and a black-box MIP solver is eventually run—in its default mode—by taking this list into account, thus avoiding any other interference with its highly-optimized internal mechanisms. Computational results on a large set of instances from MIPLIB 2010 are presented, showing that some speedup can be achieved even with respect to a state-of-the-art solver such as `IBM ILOG Cplex` 12.2.

1 Introduction

Consider a generic Mixed-Integer linear Program (MIP) of the form:

$$
\begin{aligned}
(P)\quad v(P) := \ & \min c^T x && (1)\\
& Ax \geq b, && (2)\\
& x_j \text{ integer}, \ \forall j \in \mathcal{I}, && (3)\\
& x_j \text{ continuous}, \ \forall j \in \mathcal{C}, && (4)
\end{aligned}
$$

where A is an $m \times n$ input matrix, and b and c are input vectors of dimension m and n, respectively. The variable index set $\mathcal{N} := \{1, \ldots, n\}$ is partitioned into $(\mathcal{I}, \mathcal{C})$, where $\mathcal{I}$ is the index set of the integer variables, while $\mathcal{C}$ indexes the continuous variables, if any. Bounds on the variables are assumed to be part of system (2). Removing the integrality requirement on the integer variables leads to the LP relaxation $\min\{c^T x : x \in P\}$ where $P := \{x \in \mathcal{R}^n : Ax \geq b\}$.

With a little abuse of terminology, in what follows we will say that a point x is *integer* if x_j is integer for all $j \in \mathcal{I}$ (no matter the value of the other components), *fractional* otherwise.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 183–191, 2011.

The exact solution of (P) is typically performed by using an enumeration scheme where certain branching variables are used to partition the solution set in such a way that all the leaf nodes have an LP relaxation with an optimal vertex x^* that satisfies the following *certificate condition*: x^* is either integer, or $c^T x^* \geq v(P)$—the LP cost being $+\infty$ in case of infeasibility. A branching tree whose leaves satisfy the certificate condition is called a *certificate tree*. For the sake of simplicity, in the above definition we consider a pure branch-and-bound scheme, i.e., cut generation as well as variable fixing and node preprocessing are not taken into account when checking the certificate conditions.

The computational effort for solving (P) depends on the number of nodes of the certificate tree, that in turn depends on the number of branching variables involved. So, it makes sense to define the *compactness* of a branching tree as the number of distinct branching variables associated to it—the fewer the variables, the more compact the tree. Note however that this measure is not perfect, in the sense that not all tree nodes need to be evaluated explicitly, hence the final computational effort also depends on the tree "shape" and not just on the number of involved variables. In addition, branching on a general integer (as opposed to binary) variable might not fix its fractionality, hence these variables should be counted, e.g., in a way proportional to the logarithm of their range.

A set of variables leading to a compact certificate tree is called a *backdoor* in the AI community [8]. As a matter of fact, different backdoor definitions are possible; the one we considered in the present paper is strictly related to the concept of strong backdoor in [8]. In [5] backdoors are applied to optimization problems, and the backdoor size for some problems from MIPLIB 2003 is reported for the first time. The outcome of this analysis is that, for some problems, fixing values for a very small fraction of the decisional variables is enough to easily solve the problem to optimality.

Our first order of business is to define a model that allows us to exactly compute the minimum size of a set of integer variables leading to a certificate tree. To this end, we first compute the optimal MIP value, $v(P)$, and then treat all the vertices x of P with $c^T x < v(P)$ as *obstacles* whose fractionality has to be *covered* by the branching variable set. This approach is reminiscent of Chvátal *resolution search* [4], where obstacles are associated with minimal sets of integer variables whose fixing leads to infeasibility.

Our second step is more ambitious: we try to use backdoor information on the fly, within the solution procedure, so as to reduce the total computational effort of the MIP solver at hand. As a simple proof-of-concept of the potential of the idea, we implemented the following multiple restart scheme. We make a sequence of short enumeration runs in a "sampling mode" intended to gather information relevant for branching. This approach is in the spirit of the recent work in [6], but in our case we intend to discover and store a collection of relevant low-cost fractional solutions qualifying as obstacles that block the lower bound improvement. More specifically, we use a dynamically-updated cost threshold, and maintain a list of fractional solutions whose cost does not exceed the current threshold. At each sampling run, we start by solving a set covering model to

determine a small-cardinality set of branching variables "covering" all fractional solutions in our current list, and treat them as first-level variables with priority 1 for branching (all other variables having priority 0). We then run the MIP solver by using the above branching priorities, and collect additional low-cost fractional solutions. After a while, we abort the current run, and repeat. In this way, more and more low-cost fractional solutions are collected and used to find a clever set of first-level branching variables. After a certain number of restarts, the "final run" is executed by using the MIP solver as a black box, without interfering with its highly-optimized criteria but just giving branching priority 1 to all the variables in the solution of the last set covering problem, and 0 to the remaining ones.

Computational results on a set of 61 hard instances from MIPLIB 2010 [7] (possibly involving general-integer variables, treated by our set covering model in a heuristic way) are reported, with a comparison with IBM ILOG Cplex 12.2. The outcome is that even a simple implementation of backdoor branching can lead to a significant performance improvement.

2 A Basic Set Covering Model

We next show how to compute the *compactness* of a given MIP, defined as size of its minimum-cardinality backdoor—i.e., as the minimum number of branching variables leading to a certificate tree. To have a more meaningful setting, in this section we restrict our attention to 0-1 MIPs, i.e., all integer variables are assumed to be binary, hence the smaller the number of variables in the backdoor, the smaller the associated (complete) certificate tree—assuming again that all the nodes of this tree need to be generated explicitly.

Branching on 0-1 variables (or, more generally, on faces of P) has the nice property of not introducing new vertices of P; see, e.g., [2]. The key property here is that branching could in fact be implemented by just adding (resp. subtracting) a large constant $M > 0$ to the cost c_b of all branching variables fixed to 0 (resp., to 1), so the LP relaxation polyhedron at any node does not change. More formally, if x^k is a vertex of the LP relaxation polyhedron P^k (say) at any given branching node k, then there exists an objective function $w^k x$ that attains its unique minimum over P^k at x^k. By adding/subtracting M to the weight w^k_b of the branching variables x_b along the path from the root to node k, we then obtain a new weight function $\overline{w}^T x$ that has x^k as its unique minimum with respect to P, hence x^k is also a vertex of P.

In view of the above property, we can model our compactness problem as follows. For each $j \in \mathcal{I}$, we introduce a binary variable such that $y_j = 1$ is x_j if in the minimum backdoor set, $= 0$ otherwise. For any vertex x^* of P, let $frac(x^*) := \{j \in \mathcal{I} : x^*_j \text{ is fractional}\}$ denote its *fractional support* w.r.t. $\mathcal{I}$. The problem of computing MIP compactness then calls for a minimum-cardinality set of variables that cover all the fractionalities of all the vertices of P having a cost strictly less than $v(P)$, and can be rephrased as the following set covering problem.

$$\min \sum_{j \in \mathcal{I}} \gamma_j y_j \tag{5}$$

$$\sum_{j \in frac(x^k)} y_j \geq 1, \ \forall \text{ vertex } x^k \text{ of } P : c^T x^k < v(P) \tag{6}$$

$$y_j \in \{0,1\} \ \forall j \in \mathcal{I}, \tag{7}$$

where we set $\gamma_j = 1$ for all $j \in \mathcal{I}$, though a different cost definition can be used to obtain approximate backdoors; see below. Whenever (6) holds for a certain pair y and x^k, we say that "y covers the fractionality of x^k".

The set covering above can be tackled through the iterative generation of the covering constraints (6). To this end, a possible Benders-like solution scheme can be sketched as follows. We iteratively solve a relaxation of the set covering model where constraints (6) are written for a subset of vertices x^k, while the integrality condition (7) is preserved. For any (almost-) optimal solution $y^* \in \{0,1\}^{\mathcal{I}}$, our order of business is to prove that the complete branching tree associated with y^* is in fact a certificate tree. This in turn amounts to solving a relaxed MIP obtained from the original one by relaxing the integrality requirement for all x_j with $y_j = 0$. If one or more fractional optimal solutions x^k of this relaxed MIP are found, and $c^T x^k < v(P)$, the associated constraints (6) are added to the set covering model and the method is iterated.

3 Backdoor Branching

As already mentioned, our *backdoor branching* is a multi-restart strategy inspired by the set covering model outlined in the previous section. Its goal is to heuristically use the information associated with the set covering model (5)-(7) to produce a solution scheme for the original MIP that can outperform the standard one, at least on certain classes of hard instances.

Our underlying hypothesis here is that even hard MIPs could be solved by a compact tree (in terms of number of branching variables involved), if an appropriate set of branching variables is chosen. So, we afford spending a certain amount of computing time just to heuristically determine a small set of branching variables leading to a (possibly approximate) certificate tree.

Our proposal can be cast into the following master-slave scheme.

The set covering problem (5)-(7) written for a certain set of vertices x^k acts as our *master problem*: it is solved (possibly heuristically) with a short time limit to get a candidate branching set y^*.

If $\sum_{j \in \mathcal{I}} y_j^*$ is reasonably small, then the branching set is compact and we try to use it. To be specific, we solve our MIP as a *slave problem* to possibly generate new x^k's needed to extend (6). To this end, instead of relaxing the integrality conditions on the x_j's with $y_j^* = 0$ we prefer to use a black-box MIP solver on the *original* problem (P) by giving all variables x_j with $y_j^* = 1$ a high priority for branching. This is obtained by just using y^* as a branching priority vector

(=1 for the variables to branch first, =0 for the other variables to be branched only as a last resort), so as to let the MIP-solver choose among these variables according to its favorite strategy, e.g., strong branching or alike.

All fractional solutions x^k encountered during the solution of the slave problem are stored and added to the list in (6) to be possibly used at the next iteration. In order to favor the collection of low-cost solutions x^k, a best-bound-first tree exploration strategy is used.

When a prefixed number of new solutions x^k whose fractionality is not covered by y^* has been collected (say, $K_{max} = 10$) we have the evidence that the current y^* is not an accurate estimate of a good branching set, and we return to the master to get another y^*, i.e., a new tentative branching set.

The master-slave cycle above is iterated until either (i) too many slaves have been solved (10, in our implementation), or (ii) the current master solution y^* has too many 1's (more than 5, in our implementation), meaning that the MIP likely does not admit a compact branching set. In the latter case, instead of giving up we relax the requirement of having an exact backdoor, and reduce the threshold value of the vertices x^k to be taken into account in (6). In particular, denoting by θ_c and $v(LP)$ the current threshold value and the value of the LP relaxation, respectively, we reduce the threshold by setting

$$\theta_c = \theta_c - (\theta_c - v(LP))/10.$$

At the first iteration, θ_c is initialized to the value of the best integer solution known.

After the solution of the last slave problem, we make the final choice of the branching priorities by solving our last set covering model (5)-(7) over the current set of fractional solutions x^k whose cost is smaller than the current θ_c, and just re-run the MIP solver from scratch (in its default setting) by using the optimal set covering solution y^* as priority vector—this phase being called "the final run".

As the solution of the original MIP (acting as a slave in our scheme) is restarted several times, our method can be interpreted as a multi-restart strategy under the control of a set covering model (the master) that guides the branching process.

Due to its heuristic nature, the backdoor definition above can be quite risky in practice because it may favor variables that happen to cover well the collected fractionalities, but have a minor impact for branching. An extreme case arises when the backdoor includes variables whose integrality requirement is in fact redundant. To mitigate this drawback, we try to exploit the information available for free after each slave solution. To this end, we count how many times each variable has been selected for branching in any of the slaves, and also store the pseudocost vector (see, e.g., [3] and [1]) when the slave is interrupted. Each time a set covering is solved, we treat the never-branched variables as if they were continuous, and modify the set covering objective function to favor the choice of the variables that are likely to have a large impact for branching. To be specific, the cost of each variable y_j ($j \in \mathcal{I}$) in model (5)–(7) is redefined

as $\gamma_j = M - p_j$, where M is a sufficiently large number, and p_j measures the importance of variable j for branching (the larger the more important). In our proof-of-concept implementation, we set $M = 1000$, whereas coefficient p_j is computed as

$$p_j = \lfloor 100 \cdot \frac{\Psi_j^- + \Psi_j^+}{\Psi_{\max}} \rfloor \tag{8}$$

where Ψ_j^- and Ψ_j^+ denote the average (across all slaves solved so far) pseudocost of variable j in the downwards and upwards branching, respectively, and $\Psi_{\max} = \max\{\Psi_j^- + \Psi_j^+ : j \in \mathcal{I}\}$. Different definitions of the p_j's are also possible, that take into account how many times a variable has been selected for branching.

4 Preliminary Computational Results

In this section we compare our backdoor branching scheme with the state-of-art MIP solver `IBM ILOG Cplex` 12.2. All experiments were performed on an Intel(R) Xeon(R) CPU E5450 running at 3.0 GHz, in single-thread mode, with a time-limit of 10,000 CPU seconds for each run.

Our testbed was constructed to contain "not too easy nor hopeless" instances, selected according to the following procedure intended to reduce biasing in favor of one of two codes under comparison.

We first considered all MIPLIB 2010 [7] instances and solved them by `IBM ILOG Cplex` 12.2 (in its default setting, with no upper cutoff). Then, we disregarded the instances that could not be solved in the time limit, along with the "too easy" ones that could be solved within just 1,000 nodes. This produces a first testbed made by 212 problems.

Following [6], to have a fair comparison of alternative branching rules with no side effects related to primal heuristics, in the subsequent runs we provided the optimal solution value as the upper cutoff to both solvers and disabled all heuristics, which became useless in this context. In this new framework, some instances in our testbed became "too easy" and were discarded, resulting in our final testbed of 98 instances.

Again to reduce side effects, in subsequent runs we deactivated cut generation for both codes—resulting into a significant slowdown or even a time-limit condition for some instances. In addition, we also had to deactivate (again for both codes) variable aggregation in the preprocessing phase because of its interference with the branching priority mechanism.

Tables 1 and 2 provide the outcome of our experiments on the 61 instances in our testbed that could be solved to proven optimality by at least one of the two codes under comparison. For each instance we report, for both `IBM ILOG Cplex` 12.2 and backdoor branching, the computing time and number of branching nodes (`t.l.` indicates that the instance was not solved to proven optimality within the 10,000-second time limit). Computing times and number of nodes reported for our backdoor method sum up the corresponding figures for all (sampling and final) runs; the root node computing time is instead counted only once in that all these runs start exactly from the same root-node information. In addition,

we report the size $|B|$ of the "approximate backdoor" used for the last run of the MIP solver, as well as the computing time (column T_{last}) spent by backdoor branching in its final run.

The last lines of the table report the average values (arithmetic and geometric means) of computing time and number of nodes, along with the number of time-limit instances. Unsolved instances are counted by taking the computing time and number of nodes at the time limit.

Though preliminary, the results show that a significant speedup can be obtained by backdoor branching even with respect to a highly sophisticated and

Table 1. Detailed results for `IBM ILOG Cplex` 12.2 and backdoor branching on a set of hard instances, when the optimal value is provided on input and heuristics and cuts are deactivated. `t.l.` indicates that the 10,000-sec. time limit has been reached.

	Cplex		Backdoor			
	Time	#nodes	Time	#nodes	$\|B\|$	T_{last}
018_aflow40b	5,939.95	4,253,462	5,334.50	3,185,782	3	5,331.03
021_app1_2	323.36	6,064	791.83	12,224	2	698.76
040_bienst2	87.73	85,199	153.15	143,731	4	152.46
060_csched-010	3,010.52	2,174,057	3,620.00	2,469,581	5	3,618.61
081_eilD76.2	2,945.78	135,801	1,904.51	119,656	3	1,833.33
111_glass.lp.sc	2,986.62	131,975	1,011.72	44,897	2	967.86
112_gmu35_40	t.l.	63,137,971	90.04	439,918	5	89.72
161_markshare_5_0	4,900.52	141,061,124	4,479.73	122,768,127	6	4,479.55
166_mcsched	560.68	72,102	701.75	162,209	2	685.42
172_mine_90_10	823.09	217,682	1,741.71	436,330	2	1,716.47
175_miplib1	2,803.23	16,086	3,425.80	16,218	3	2,930.37
209_neos-1126860	4,279.68	18,519	3,437.78	15,993	5	3,310.87
216_neos-1215891	1,860.27	442,404	809.99	178,128	3	798.94
222_neos-1330346	t.l.	562,829	9,307.20	532,478	2	9,289.45
223_neos-1337307	8,127.90	475,838	t.l.	380,415	5	t.l.
234_neos-1427181	t.l.	4,770,447	2,619.10	934,040	6	2,611.67
239_neos-1439395	24.04	16,866	149.60	109,735	10	147.04
242_neos-1440460	2,348.67	1,657,377	958.45	764,126	6	958.03
246_neos-1451294	8,968.85	71,278	t.l.	91,796	3	t.l.
255_neos-1595230	311.47	79,340	267.76	66,921	3	264.20
262_neos-1616732	6,847.04	2,747,954	4,538.61	1,996,385	2	4,529.23
267_neos18	211.50	143,654	174.34	122,214	3	170.28
280_neos-548047	t.l.	106,331	6,312.55	130,069	2	6,261.13
294_neos5	145.34	1,199,529	60.42	447,350	3	59.83
295_neos-603073	2.31	4,983	3.08	6,238	3	2.64
313_neos-785912	t.l.	614,039	3,517.74	242,110	4	3,503.88
315_neos788725	218.60	767,350	201.03	755,774	3	199.55
322_neos818918	31.00	7,003	148.90	38,194	3	143.13
326_neos823206	365.19	205,241	255.14	167,653	3	250.07
345_neos-859770	3,030.08	147,225	1,778.74	67,867	6	1,762.53
347_neos-863472	17.03	56,334	30.59	102,266	2	30.40

Table 2. (*continued*)

	Cplex		Backdoor			
	Time	#nodes	Time	#nodes	$\|B\|$	T_{last}
353__neos-886822	67.43	15,360	61.20	15,383	2	52.97
359__neos-916792	443.46	94,039	542.16	121,539	2	535.58
403__noswot	222.96	2,353,215	112.79	1,154,341	8	112.67
413__ns1324654	5,644.19	52,413	t.l.	153,361	3	t.l.
433__ns1702808	290.14	251,583	407.94	377,091	5	407.06
438__ns1766074	152.87	1,212,703	157.33	1,275,784	5	156.68
468__ns25-pr3	1,376.72	219,964	1,384.56	207,664	4	1,379.16
469__ns25-pr9	134.02	23,230	87.58	15,752	3	84.46
472__ns60-pr3	t.l.	1,342,213	7,687.11	1,066,696	2	7,679.68
473__ns60-pr9	246.01	38,056	256.52	38,491	2	250.49
496__opm2.z8.s1_frm00	3,181.25	10,578	5,234.70	11,564	2	4,134.92
501__p2m2p1m1p0n100	1,461.60	64,618,970	1,287.10	64,620,437	30	1,286.83
506__pdh-DBM	15.12	56,602	16.13	57,454	2	15.56
510__pigeon-09	57.25	467,325	74.34	621,353	3	74.10
511__pigeon-10	657.46	4,594,047	670.62	4,676,256	3	670.08
516__pima.lp.sc	737.38	17,850	508.45	9,728	2	453.94
518__prob.15.80.100.4.sc	7,062.42	195,991	7,510.51	187,229	2	7,411.97
519__prob.20.90.100.0.sc	2,032.45	23,035	1,996.37	20,743	2	1,812.63
520__prob.25.80.100.1.sc	1,781.13	8,050	3,707.84	10,279	2	3,127.18
521__prob.25.90.100.2.sc	3,214.90	45,809	4,103.90	45,746	2	3,835.31
522__prob.5.100.100.0.sc	2,110.80	185,749	1,528.52	162,617	2	1,504.69
523__prob.5.100.100.3.sc	4,028.78	270,359	1,898.21	139,086	2	1,860.27
527__pw-myciel4	185.88	26,842	382.78	60,610	7	376.52
544__ran16x16	2,048.57	14,953,792	2,289.32	16,702,697	4	2,289.12
565__rocII_4_11	484.77	158,374	409.18	124,170	5	401.23
571__rococoC10-001000	2,061.76	1,631,422	3,019.17	1,608,535	2	3,016.19
601__SING290	t.l.	237,926	7,850.87	123,885	3	7,706.96
608__sp98ic	746.97	528,380	790.75	536,194	3	780.00
634__tic-tac-toe.lp.sc	3,136.55	52,866	2,674.17	54,071	2	2,585.25
655__wpbc.lp.sc	5,432.18	37,310	4,454.31	44,400	2	4,245.28
arithmetic mean	2,953.86	5,231,313	2,443.72	3,790,026		2,377.76
geometric mean	961.25	215,234	871.64	193,611		844.23
number of t.l.	7		3			

effective commercial solver such as `IBM ILOG Cplex` 12.2, at least in our setting. In particular, the backdoor branching strategy could solve 4 more instances than the standard strategy, with a global speedup of about 10% in geometric mean, and 20% in arithmetic mean. The number of branching nodes decreased by about 10% in geometric mean, and by about 40% in arithmetic mean. An interesting outcome of our experiments is that approximate backdoors of very small size may be quite useful in shaping the enumeration tree in a more effective way.

We also performed very preliminary experiments on a variant of our backdoor branching where the MIPs solved during the sampling phase use strong branching for a more careful choice of the branching variables, thus making the slave

information more reliable and useful for the backdoor choice. This policy turned out to be rather effective in reducing the time spent in the final run, though this did not compensate for the sampling time increase due to strong branching, at least in our current implementation.

Acknowledgments

Research was supported by the *Progetto di Ateneo* on "Computational Integer Programming" of the University of Padova.

References

1. Achterberg, T.: Constraint Integer Programming. PhD thesis, Technische Universität Berlin; Fakultät II - Mathematik und Naturwissenschaften. Institut für Mathematik (2007)
2. Balas, E., Ceria, S., Cornuéjols, G., Natraj, N.: Gomory cuts revisited. Operations Research Letters 19, 1–9 (1996)
3. Bénichou, M., Gauthier, J.M., Girodet, P., Hentges, G., Ribière, G., Vincent, O.: Experiments in mixed integer linear programming. Mathematical Programming 1, 76–94 (1971)
4. Chvátal, V.: Resolution search. DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science 73 (1997)
5. Dilkina, B., Gomes, C.P., Malitsky, Y., Sabharwal, A., Sellmann, M.: Backdoors to Combinatorial Optimization: Feasibility and Optimality. In: van Hoeve, W.-J., Hooker, J.N. (eds.) CPAIOR 2009. LNCS, vol. 5547, pp. 56–70. Springer, Heidelberg (2009)
6. Karzan, F.K., Nemhauser, G.L., Savelsbergh, M.W.P.: Information-based branching schemes for binary linear mixed integer problems. Mathematical Programming Computation 1, 249–293 (2009)
7. MIPLIB 2010. Preliminary version, http://miplib.zip.de/
8. Williams, R., Gomes, C.P., Selman, B.: Backdoors to typical case complexity. In: Gottlob, G., Walsh, T. (eds.) IJCAI 2003: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 1173–1178. Morgan Kaufmann, San Francisco (2003)

A Subexponential Lower Bound for Zadeh's Pivoting Rule for Solving Linear Programs and Games

Oliver Friedmann

Department of Computer Science,
University of Munich, Germany
Oliver.Friedmann@gmail.com

Abstract. The *simplex* algorithm is among the most widely used algorithms for solving *linear programs* in practice. Most pivoting rules are known, however, to need an exponential number of steps to solve some linear programs. No non-polynomial lower bounds were known, prior to this work, for *Zadeh's* pivoting rule [25].

Also known as the Least-Entered rule, Zadeh's pivoting method belongs to the family of memorizing improvement rules, which among all improving pivoting steps from the current basic feasible solution (or *vertex*) chooses one which has been entered least often. We provide the first *subexponential* (i.e., of the form $2^{\Omega(\sqrt{n})}$) lower bound for this rule.

Our lower bound is obtained by utilizing connections between pivoting steps performed by simplex-based algorithms and *improving switches* performed by *policy iteration* algorithms for 1-player and 2-player games. We start by building 2-player *parity games* (PGs) on which the policy iteration with the Least-Entered rule performs a subexponential number of iterations. We then transform the parity games into 1-player *Markov Decision Processes* (MDPs) which corresponds almost immediately to concrete linear programs.

1 Introduction

The *simplex method*, developed by Dantzig in 1947 (see [5]), is among the most widely used algorithms for solving linear programs. One of the most important parameterizations of a simplex algorithm is the *pivoting rule* it employs. It specifies which non-basic variable is to enter the basis at each iteration of the algorithm. Although simplex-based algorithms perform very well in practice, essentially all *deterministic* pivoting rules are known to lead to an *exponential* number of pivoting steps on some LPs [21], [18], [1] and [15].

Kalai [19,20] and Matoušek, Sharir and Welzl [22] devised *randomized* pivoting rules that never require more than an expected *subexponential* number of pivoting steps to solve any linear program. The most prominent randomized pivoting rules probably are Random-Facet [19,20,22] and Random-Edge [4,13,14], for which, until recently [12], no non-trivial lower bounds given by concrete linear programs were known.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 192–206, 2011.

An interesting deterministic pivoting rule for which no subexponential lower bound is known yet was suggested by Zadeh [25] (see also [7]). Also known as the Least-Entered rule, Zadeh's pivoting method belongs to the family of memorizing improvement rules, which among all improving pivoting steps from the current basic feasible solution (or *vertex*) chooses one which has been entered least often.

Here, we provide the first *subexponential* (i.e., of the form $2^{\Omega(\sqrt{n})}$) lower bound for the this rule.

Techniques used. The linear program on which Least-Entered performs a subexponential number of iterations is obtained using the close relation between simplex-type algorithms for solving linear programs and *policy iteration* (also known as *strategy improvement*) algorithms for solving certain 2-player and 1-player games.

This line of work was started by showing that standard strategy iteration [24] for parity games [16] may require an exponential number of iterations to solve them [9]. Fearnley [8] transfered the lower bound construction for parity games to *Markov Decision Processes* (MDPs) [17], an extremely important and well studied family of stochastic 1-player games.

In [11], we recently constructed PGs on which the Random-Facet algorithm performs an expected subexponential number of iterations. In [12], we applied Fearnley's technique to transform these PGs into MDPs, and include an additional lower bound construction for the Random-Edge algorithm.

The problem of solving an MDP, i.e., finding the optimal control *policy* and the optimal *values* of all states of the MDP, can be cast as a linear program. More precisely, the improving switches performed by the (abstract) Least-Entered algorithm applied to an MDP corresponds directly to the steps performed by the Least-Entered pivoting rule on the corresponding linear program.

Our results. We construct a family of concrete linear programs on which the number of iterations performed by Least-Entered is $2^{\Omega(\sqrt{n})}$, where n is the number of variables.

Here, we follow our approach from [12] to obtain a subexponential lower bound for Zadeh's pivoting rule by constructing concrete parity games on which the policy iteration algorithm parameterized with Zadeh's rule requires a subexponential number of iterations. Then, we transform the PGs into MDPs, and the linear programs corresponding to our MDPs supply, therefore, concrete linear programs on which following the Least-Entered pivoting rule leads to an expected subexponential number of iterations.

As the translation of our PGs to MDPs is a relatively simple step, we directly present the MDP version of our construction. As a consequence, our construction can be understood without knowing anything about PGs.

In high level terms, our PGs, MDPs, and the linear programs corresponding to them, are constructions of 'pairwise alternating' binary counters. Consider a normal binary counter: less significant bits are switched more often than higher bits, when counting from 0 to $2^n - 1$. Zadeh's rule would not go through all steps

from 0 to $2^n - 1$ on such a counter, because higher bits will be switched before they are supposed to be switched, as the switching times that are associated with higher bits will catch up with the switching times associated with lower bits. Zadeh's rule, in a sense, requires a "fair" counter that operates correctly when all bits are switched equally often.

Our solution to this problem is to represent each bit i in the original counter by *two* bits i' and i'' s.t. only one of those two is actively working as representative for i. After switching the representative for i – say i' – from 0 to 1 and back to 0, we change the roles of i' and i'' s.t. i'' becomes the active representative for i. The inactive i' can now, while i'' switches from 0 to 1 and back to 0, catch up with the rest of the counter in terms of switching fairness: while i' is inactive, we switch i' from 0 to 1 back and forth (without effecting the rest of the counter as i' is the inactive representative) until the number of switching times catches up with the number of switching times of the rest of the counter again.

Another viable approach could be to implement more sophisticated binary counters like Gray codes (see e.g. [3]). However, the construction of an MDP or PG that models the behavior of a Gray code-based counter seems to be a very difficult task.

The rest of this paper is organized as follows. In Section 2 we give a brief introduction to *Markov Decision Processes* (MDPs) and the primal linear programs corresponding to them. In Section 3 we review the policy iteration and the simplex algorithms, the relation between improving switches and pivoting steps, and Zadeh's Least-Entered pivoting rule. In Section 4, which is the main section of this paper, we describe our lower bound construction for Least-Entered. We end in Section 5 with some concluding remarks and open problems.

2 Markov Decision Processes and Their Linear Programs

Markov decision processes (MDPs) provide a mathematical model for sequential decision making under uncertainty. They are employed to model stochastic optimization problems in various areas ranging from operations research, machine learning, artificial intelligence, economics and game theory. For an in-depth coverage of MDPs, see the books of Howard [17], Derman [6], Puterman [23] and Bertsekas [2].

Formally, an MDP is defined by its *underlying graph* $G=(V_0, V_R, E_0, E_R, r, p)$. Here, V_0 is the set of vertices (states) operated by the controller, also known as *player* 0, and V_R is a set of *randomization* vertices corresponding to the probabilistic actions of the MDP. We let $V = V_0 \cup V_R$. The edge set $E_0 \subseteq V_0 \times V_R$ corresponds to the actions available to the controller. The edge set $E_R \subseteq V_R \times V_0$ corresponds to the probabilistic transitions associated with each action. The function $r : E_0 \to \mathbb{R}$ is the immediate reward function. The function $p : E_R \to [0, 1]$ specifies the transition probabilities. For every $u \in V_R$, we have $\sum_{v:(u,v)\in E_R} p(u, v) = 1$, i.e., the probabilities of all edges emanating from each vertex of V_R sum up to 1. As defined, the graph G is *bipartite*, but one can relax this condition and allow edges from V_0 to V_0 that correspond to *deterministic* actions.

A policy σ is a function $\sigma : V_0 \to V$ that selects for each vertex $u \in V_0$ a target node v corresponding to an edge $(u, v) \in E_0$, i.e. $(u, \sigma(u)) \in E_0$ wWe assume that each vertex $u \in V_0$ has at least one outgoing edge). There are several *objectives* for MDPs; we consider the *expected total reward objective* here. The *values* $\text{VAL}_\sigma(u)$ of the vertices under σ are defined as the unique solutions of the following set of linear equations:

$$\text{VAL}_\sigma(u) = \begin{cases} \text{VAL}_\sigma(v) + r(u,v) & \text{if } u \in V_0 \text{ and } \sigma(u) = v \\ \sum_{v:(u,v)\in E_R} p(u,v)\,\text{VAL}_\sigma(v) & \text{if } u \in V_R \end{cases}$$

together with the condition that $\text{VAL}_\sigma(u)$ sum up to 0 on each irreducible recurrent class of the Markov chain defined by σ.

All MDPs considered in this paper satisfy a *weak* version of the *unichain* condition. The normal unichain condition (see [23]) states that the Markov chain obtained from each policy σ has a single irreducible recurrent class. We discuss the weak version at the end of this section.

Optimal policies for MDPs that satisfy the unichain condition can be found by solving the following (primal) linear program

$$(P) \quad \begin{array}{ll} \max & \sum_{(u,v)\in E_0} r(u,v)x(u,v) \\ \text{s.t.} & \sum_{(u,v)\in E} x(u,v) - \sum_{(v,w)\in E_0,(w,u)\in E_R} p(w,u)x(v,w) = 1,\ u \in V_0 \\ & x(u,v) \ \geq \ 0 \quad , \quad (u,v) \in E_0 \end{array}$$

The variable $x(u, v)$, for $(u, v) \in E_0$, stands for the probability (frequency) of using the edge (action) (u, v). The constraints of the linear program are *conservation constraints* that state that the probability of entering a vertex u is equal to the probability of exiting u. It is not difficult to check that the *basic feasible solutions* (bfs's) of (P) correspond directly to policies of the MDP. For each policy σ we can define a feasible setting of primal variables $x(u, v)$, for $(u, v) \in E_0$, such that $x(u, v) > 0$ only if $\sigma(u) = (u, v)$. Conversely, for every bfs $x(u, v)$ we can define a corresponding policy σ. It is well known that the policy corresponding to an optimal bfs of (P) is an optimal policy of the MDP. (See, e.g., [23].)

Our MDPs only satisfy a *weak* version of the unichain condition, saying that the optimal policy has a single irreducible recurrent class. It follows that the optimal policy can be found by the same LPs when being started with an initial basic feasible solution corresponding to a policy with the same single irreducible recurrent class as the optimal policy. Then, by monotonicity, we know that all considered basic feasible solutions will have the same irreducible recurrent class.

It should be noted that *all* pivoting steps performed on these linear programs are *non-degenerate*, due to the fact the we consider the expected total reward criterion here. The lower bound construction of this paper also works when applied to the *discounted reward criterion* (for large enough discount factors), and also for the *limiting average reward criterion*. However, in the latter case, all pivoting steps performed on the induced linear programs are *degenerate*.

3 Policy Iteration Algorithms and Simplex Algorithms

Howard's [17] *policy iteration* algorithm is the most widely used algorithm for solving MDPs. It is closely related to the simplex algorithm.

The algorithm starts with some initial policy σ_0 and generates an improving sequence $\sigma_0, \sigma_1, \ldots, \sigma_N$ of policies, ending with an optimal policy σ_N. In each iteration the algorithm first *evaluates* the current policy σ_i, by computing the values $\text{VAL}_{\sigma_i}(u)$ of all vertices. An edge $(u, v') \in E_0$, such that $\sigma_i(u) \neq v'$ is then said to be an *improving switch* if and only if either $\text{VAL}_{\sigma_i}(v') > \text{VAL}_{\sigma_i}(u)$. Given a policy σ, we denote the *set of improving switches* by I_σ.

A crucial property of policy iteration is that σ is an optimal policy if and only if there are no improving switches with respect to it (see, e.g., [17], [23]). Furthermore, if $(u, v') \in I_\sigma$ is an improving switch w.r.t. σ, and σ' is defined as $\sigma[(u, v')]$ (i.e., $\sigma'(u) = v'$ and $\sigma'(w) = \sigma(w)$ for all $w \neq u$), then σ' is *strictly* better than σ, in the sense that for every $u \in V_0$, we have $\text{VAL}_{\sigma'}(u) \geq \text{VAL}_\sigma(u)$, with a strict inequality for at least one vertex $u \in V_0$.

Policy iteration algorithms that perform a single switch at each iteration – like Zadeh's rule – are, in fact, simplex algorithms. Each policy σ of an MDP immediately gives rise to a feasible solution $x(u, v)$ of the primal linear program (P); use σ to define a Markov chain and let $x(u, v)$ be the 'steady-state' probability that the edge (action) (u, v) is used. In particular, if $\sigma(u) \neq v$, then $x(u, v) = 0$.

Zadeh's Least-Entered pivoting rule is a *deterministic*, *memorizing* improvement rule which among all improving pivoting steps from the current basic feasible solution (or *vertex*) chooses one which has been entered least often. When applied to the primal linear program of an MDP, it is equivalent to the variant of the policy iteration algorithm, in which the improving switch is chosen among all improving switches to be one, which has been chosen least often. This is the foundation of our lower bound for the Least-Entered rule.

We describe Zadeh's pivoting rule now formally in the context of MDPs. As a memorization structure, we introduce an *occurrence record*, which is a map $\phi : E_0 \to \mathbb{N}$ that specifies for every player 0 edge of the given MDP how often it has been used. Among all improving switches in the set I_σ for a given policy σ, we need to choose an edge $e \in I_\sigma$ that has been selected least often. We denote the set of *least occurred improving switches* by $I^\phi_\sigma = \{e \in I_\sigma \mid \phi(e) \leq \phi(e') \text{ for all } e' \in I_\sigma\}$.

See Algorithm 1 for a pseudo-code specification of the Least-Entered pivoting rule for solving MDPs.

In the original specification of Zadeh's algorithm [25], there is no clear objective how to *break ties* whenever $|I^\phi_\sigma| > 1$. In fact, we know that the asymptotic behavior of Zadeh's improvement rule highly depends on the method that is used to break ties, at least in the world of MDPs, PGs and policy iteration for games in general. We have the following theorem which is easy to verify (the idea is that there is at least one improving switch towards the optimal policy in each step).

Algorithm 1. Zadeh's Improvement Algorithm

```
1: procedure LEAST-ENTERED(G,σ)
2:     φ(e) ← 0 for every e ∈ E_0
3:     while I_σ ≠ ∅ do
4:         e ← select edge from I_σ^φ
5:         φ(e) ← φ(e) + 1
6:         σ ← σ[e]
7:     end while
8: end procedure
```

Theorem 1. *Let G be an MDP with n nodes and σ_0 be a policy. There is a sequence policies $\sigma_0, \sigma_1, \ldots, \sigma_N$ and a sequence of* different *switches $e_1, e_2, \ldots, e_N$ with $N \leq n$ s.t. σ_{N-1} is optimal, $\sigma_{i+1} = \sigma_i[e_{i+1}]$ and e_{i+1} is an σ_i-improving switch.*

Since all switches are different in the sequence, it follows immediately that there is always a way to break ties that results in a linear number of pivoting steps to solve an MDP with Zadeh's improvement rule. However, there is no obvious method of breaking ties. The question whether Zadeh's pivoting rule solves MDPs (and LPs) in polynomial time should therefore be phrased *independently* of the heuristic of breaking ties. In other words, we as "lower bound designers" are the ones that choose a particular tie breaking rule

Formally, we write $(\sigma, \phi) \rightsquigarrow (\sigma', \phi')$ iff there is an edge $e \in I_\sigma^\phi$ s.t. $\sigma' = \sigma[e]$ and $\phi' = \phi[e \mapsto \phi(e) + 1]$. Let $\rightsquigarrow^+$ denote the transitive closure of $\rightsquigarrow$. The question, whether Zadeh's improvement rule admits a polynomial number of iterations independently of the method of breaking ties is therefore equivalent to the question, whether the length of any sequence $(\sigma_0, \phi_0) \rightsquigarrow^+ \ldots \rightsquigarrow^+ (\sigma_N, \phi_N)$ can be polynomially bounded in the size of the game.

We will not specify the tie-breaking rule used for our lower bound explicitly, due to the fact that the rule itself is not a natural one. Instead, our proof just relies on the $\rightsquigarrow$-relation, witnessing in every improvement step that we only select an improving switch that has been applied least often.

4 Lower Bound for LEAST-ENTERED

We start with a high-level description of the MDPs on which LEAST-ENTERED performs an expected subexponential number of iterations. As mentioned in the introduction, the construction may be seen as an implementation of a 'fair' counter. A schematic description of the lower bound MDPs is given in Figure 1. Circles correspond to vertices of V_0, i.e., vertices controlled by player 0, while small rectangles correspond to the randomization vertices of V_R. We use the notation $k_{[i;j]}$ to indicate that player 0 in fact has edges to every node k_l with $i \leq l \leq j$.

The MDP of Figure 1 emulates an n-bit counter. It is composed of n identical levels, each corresponding to a single bit of the counter. The i-th level ($i = 1 \ldots n$)

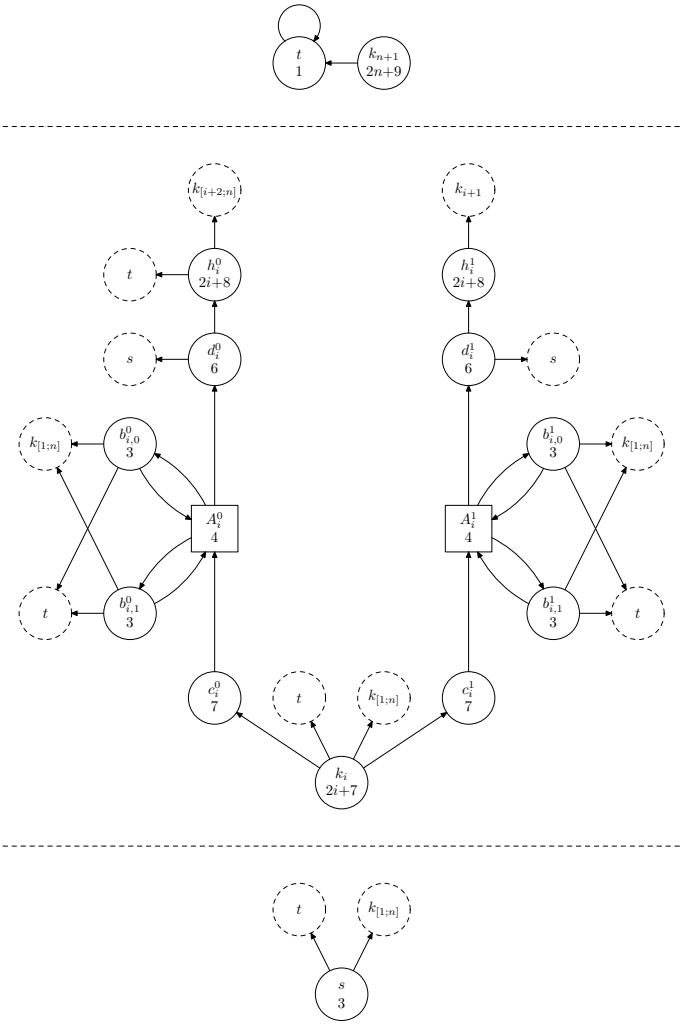


Fig. 1. Least Entered MDP Construction

is shown explicitly in the figure. Levels are separated by dashed lines. The MDP includes one *source* s and one *sink* t.

All edges in Figure 1 have an immediate reward of 0 associated with them (such 0 rewards are not shown explicitly in the figure) unless stated otherwise as follows: Some of the vertices are assigned integer *priorities*. If a vertex v has priority $\Omega(v)$ assigned to it, then a reward of $\langle v \rangle = (-N)^{\Omega(v)}$ is *added* to all edges emanating from v, where N is a sufficiently large integer. We use $N \geq 7n+1$ and $\varepsilon \leq N^{-(2n+11)}$. Priorities, if present, are listed next to the vertex name. Note that it is profitable for the controller, to move through vertices of even priority and to avoid vertices of odd priority, and that vertices of higher

numerical priority dominate vertices of lower priority (the idea of using priorities is inspired, of course, by the reduction from parity games to mean payoff games).

Each level i contains two (i.e. $j = 0, 1$) instances of a gadget that consists of a randomization vertex A_i^j and two (i.e. $l = 0, 1$) attached cycles with player 0 controlled nodes $b_{i,l}^j$. Therefore, we will call these gadgets from now on *bicycle gadgets*, and refer to the instance with $j = 0$ resp. $j = 1$ as to *bicycle* 0 resp. *bicycle* 1.

From A_i^j (with $j = 0, 1$), the edge $\mathrm{A}_i^j \to b_{i,l}^j$ (with $l = 0, 1$), is chosen with probability $\frac{1-\varepsilon}{2}$, while the edge $\mathrm{A}_i^j \to d_i^j$ is chosen with probability ε. Thus, if both $\sigma(b_{i,0}^j) = \mathrm{A}_i^j$ and $\sigma(b_{i,1}^j) = \mathrm{A}_i^j$, the MDP is guaranteed to eventually move from A_i^j to d_i^j (this is similar to the use of randomization by Fearnley [8]). We say that a bicycle gadget is

- *closed* iff both $\sigma(b_{i,0}^j) = \mathrm{A}_i^j$ and $\sigma(b_{i,1}^j) = \mathrm{A}_i^j$,
- *open* iff $\sigma(b_{i,0}^j) \neq \mathrm{A}_i^j$ or $\sigma(b_{i,1}^j) \neq \mathrm{A}_i^j$, and
- *completely open* iff $\sigma(b_{i,0}^j) \neq \mathrm{A}_i^j$ and $\sigma(b_{i,1}^j) \neq \mathrm{A}_i^j$.

Next, we introduce notation to succinctly describe binary counters. It will be convenient for us to consider counter configurations with an *infinite* tape, where unused bits are zero. The set of n-bit configurations is formally defined as $\mathcal{B}_n = \{\mathfrak{b} \in \{0,1\}^\infty \mid \forall i > n : \mathfrak{b}_i = 0\}$.

We start with index one, i.e. $\mathfrak{b} \in \mathcal{B}_n$ is essentially a tuple $(\mathfrak{b}_n, \ldots, \mathfrak{b}_1)$, with $\mathfrak{b}_1$ being the least and $\mathfrak{b}_n$ being the most significant bit. By $\mathbf{0}$, we denote the configuration in which all bits are zero, and by $\mathbf{1}_n$, we denote the configuration in which the first n bits are one. We write $\mathcal{B} = \bigcup_{n>0} \mathcal{B}_n$ to denote the set of all counter configurations.

The *integer value* of a $\mathfrak{b} \in \mathcal{B}$ is defined as usual, i.e. $|\mathfrak{b}| := \sum_{i>0} \mathfrak{b}_i \cdot 2^{i-1} < \infty$. For two $\mathfrak{b}, \mathfrak{b}' \in \mathcal{B}$, we induce the lexicographic linear ordering $\mathfrak{b} < \mathfrak{b}'$ by $|\mathfrak{b}| < |\mathfrak{b}'|$. It is well-known that $\mathfrak{b} \in \mathcal{B} \mapsto |\mathfrak{b}| \in \mathbb{N}$ is a bijection. For $\mathfrak{b} \in \mathcal{B}$ and $k \in \mathbb{N}$ let $\mathfrak{b} + k$ denote the unique $\mathfrak{b}'$ s.t. $|\mathfrak{b}'| = \mathfrak{b} + k$. If $k \leq |\mathfrak{b}|$, let $\mathfrak{b} - k$ denote the unique $\mathfrak{b}'$ s.t. $|\mathfrak{b}'| + k = |\mathfrak{b}|$.

Given a configuration $\mathfrak{b}$, we access the *i-next set bit* by $\nu_i^n(\mathfrak{b}) = \min(\{n+1\} \cup \{j \geq i \mid \mathfrak{b}_j = 1\})$, and the *$i$-next unset bit* by $\mu_i(\mathfrak{b}) = \min\{j \geq i \mid \mathfrak{b}_j = 0\}$.

The i-th level of the MDP corresponds to the i-th bit. A set bit is represented by a *closed* bicycle gadget. Every level has two bicycle gadgets, but only one of them is *actively* representing the i-th bit.

Whether bicycle 0 or bicycle 1 is active in level i depends on the setting of the i+1-th bit. If it is set, i.e. $\mathfrak{b}_{i+1} = 1$, then bicycle 1 is active in the i-th level; otherwise, if $\mathfrak{b}_{i+1} = 0$, we have that bicycle 0 is active in the i-th level.

Our proof is conceptually divided into two parts. First we investigate the improving switches that can be performed from certain policies of the MDP. This allows us to prove the *existence* of a sequence of improving switches that indeed generates the sequence of policies $\sigma_{0\ldots00}, \sigma_{0\ldots01}, \sigma_{0\ldots10}, \ldots, \sigma_{1\ldots11}$. A transition from $\sigma_{\mathfrak{b}}$ to $\sigma_{\mathfrak{b}+1}$ involves many intermediate improvement steps. We partition the path leading from $\sigma_{\mathfrak{b}}$ to $\sigma_{\mathfrak{b}+1}$ into six sub-paths which we refer to as *phases*. In the following, we first give an informal description of the phases. The second

part of our proof will be to show that the way we want to apply the improving switches is compliant with the associated occurrence records.

Before starting to describe what happens in the different phases, we describe the "ideal" configuration of a policy, which belongs to phase 1: (1) all active bicycles corresponding to set bits are closed, (2) all other bicycles are completely open, moving to the least set bit, (3) all entry points k_i move to the active bicycle if bit i is set and to the least set bit otherwise, (4) the source s moves to the least set bit, (5) all upper selection nodes h_i^0 move to the next *accessible* set bit (i.e. to the next set bit with index $\geq i{+}2$), and (6) the selection nodes d_i^j move higher up iff the immediately accessed bit is the next set bit (i.e. d_i^0 moves higher up iff $\mathfrak{b}_{i+1} = 0$ and d_i^1 moves higher up iff $\mathfrak{b}_{i+1} = 1$).

Note that the two upper selection nodes h_i^0 and h_i^1 cannot select the same entry points. The left node, h_i^0, can select from the entry points k_{i+2} up to k_n, while the right node, h_i^1, can only move to k_{i+1}. The intuition behind this is that bit $i{+}1$ is set every second time bit i is flipped, resulting in the alternating activation of the two bit representatives for i.

Now, we are ready to informally describe all phases.

1. At the beginning of the first phase, we only have open bicycles that are competing with each other to close. Inactive bicycles may have to catch up with active bicycles, and hence, are allowed to switch both player 0 edges inward, and therefore close the gadget. All active open bicycles move exactly one edge inward in this phase.
   So far, no active open bycycles have been closed. The last switch that is performed in this phase is to move the remaining edge of the active bicycle associated with the least unset bit inward, and therefore close the gadget.
2. In this phase, we need to make the recently set bit i accessible by the rest of the MDP, which will be via the k_i node. We switch here from k_i to c_i^j, where j denotes the active representative in this level.
   Note that k_i now has the highest value among all other k_*. Note that generally, k_l has a higher value than k_z for a set bit l and an unset bit z, and that k_l has a higher value than k_z for two set bits l and z iff $l < z$.
3. In the third phase, we perform the major part of the *resetting* process. By resetting, we mean to unset lower bits again, which corresponds to reopening the respective bicycles.
   Also, we want to update all other inactive or active but not set bicycles again to move to the entry point k_i. In other words, we need to update the lower entry points k_z with $z < i$ to move to k_i, and the bicycle nodes $b_{z,l}^j$ to move to k_i.
   We apply these switches by first switching the entry node k_z for some $z < i$, and then the respective bicycle nodes $b_{z,l}^j$.
4. In the fourth phase, we update the upper selection nodes h_z^0 for all $z < i - 1$ of the bits that have been reset. All nodes h_z^0 should move to k_i.
5. In the fifth phase, we update the source node to finally move to the entry point corresponding to the recently set bit i.
6. In the last phase, we only have to update the selection nodes d_z^j for all $z < i$ of the bits that have been reset. We finally end up in a phase 1 policy again with the counter increased by one.

4.1 Full Construction

In this subsection, we formally describe the full construction of our MDPs. We define an underlying graph $G_n = (V_0, V_R, E_0, E_R, r, p)$ of an MDP as shown schematically in Figure 1 as follows:

$$V_0 := \{b^0_{i,0}, b^1_{i,0}, b^0_{i,1}, b^1_{i,1}, d^0_i, d^1_i, h^0_i, h^1_i, c^0_i, c^1_i \mid i \in [n]\} \cup \{k_i \mid i \in [n+1]\} \cup \{t, s\}$$
$$V_R := \{\mathrm{A}^0_i, \mathrm{A}^1_i \mid i \in [n]\}$$

With G_n, we associate a large number $N \in \mathbb{N}$ and a small number $0 < \varepsilon$. We require N to be at least as large as the number of nodes with priorities, i.e. $N \geq 7n+1$ and ε^{-1} to be significantly larger than the largest occurring priority induced reward, i.e. $\varepsilon \leq N^{-(2n+11)}$. Remember that node v having priority $\Omega(v)$ means that the cost associated with every outgoing edge of v is $\langle v \rangle = (-N)^{\Omega(v)}$.

Table 1 defines the edge sets, the probabilities, the priorities and the immediate rewards of G_n (note that h^0_i has the successors t, k_{i+2}, ... k_n; particularly, h^0_n has only t as successor).

Table 1. Least Entered MDP Construction

Node	Successors	Probability
A^j_i	d^j_i	ε
	$b^j_{i,0}$	$\frac{1}{2} \cdot (1-\varepsilon)$
	$b^j_{i,1}$	$\frac{1}{2} \cdot (1-\varepsilon)$

Node	Successors	Priority
t	t	-
s	$t, k_{[1;n]}$	-

Node	Successors	Priority
k_{n+1}	t	$2n{+}9$
k_i	$c^0_i, c^1_i, t, k_{[1;n]}$	$2i{+}7$
h^0_i	$t, k_{[i+2;n]}$	$2i{+}8$
h^1_i	k_{i+1}	$2i{+}8$
c^j_i	A^j_i	7
d^j_i	h^j_i, s	6
$b^j_{i,*}$	$t, \mathrm{A}^j_i, k_{[1;n]}$	-

As designated *initial policy* σ^*, we use $\sigma^*(d^j_i) = h^j_i$, and $\sigma^*(\_) = t$ for all other player 0 nodes with non-singular out-degree. It is not hard to see that, starting with this initial policy, the MDP satisfies the weak unichain condition.

Lemma 1. *The Markov chains obtained by the initial and the optimal policy reach the sink t almost surely (i.e. the sink t is the single irreducible recurrent class).*

It is not too hard to see that the absolute value of all nodes corresponding to policies belonging to the phases are bounded by ε^{-1}. More formally we have:

Lemma 2. *Let $P = \{k_*, h^*_*, c^*_*, d^*_*\}$ be the set of nodes with priorities. For a subset $S \subseteq P$, let $\sum(S) = \sum_{v \in S} \langle v \rangle$. For non-empty subsets $S \subseteq P$, let $v_S \in S$ be the node with the largest priority in S.*

1. *$|\sum(S)| < N^{2n+11}$ and $\varepsilon \cdot |\sum(S)| < 1$ for every subset $S \subseteq P$, and*
2. *$|v_S| < |v_{S'}|$ implies $|\sum(S)| < |\sum(S')|$ for non-empty subsets $S, S' \subseteq P$.*

4.2 Lower Bound Proof

In this subsection, we formally describe the different *phases* that a policy can be in, as well as the improving switches in each phase. The increment of the binary counter by one is realized by transitioning through all the phases. Finally, we describe the corresponding occurrence records that appear in a run of the policy iteration on the MDPs.

We first introduce notation to succinctly describe policies. It will be convenient to describe the decision of a policy σ in terms of integers rather than concrete target vertices. Let σ be a policy. We define a function $\bar{\sigma}(v)$ as follows.

$\sigma(v)$	t	k_i	h^*_*	s	A^*_*	c^j_i
$\bar{\sigma}(v)$	$n+1$	i	1	0	0	$-j$

Additionally, we write $\bar{\sigma}(\mathrm{A}^j_i) = 1$ if $\sigma(b^j_{i,0}) = \mathrm{A}^j_i$ and $\sigma(b^j_{i,1}) = \mathrm{A}^j_i$, and $\bar{\sigma}(\mathrm{A}^j_i) = 0$ otherwise.

We are now ready to formulate the conditions for policies that fulfill one of the six phases along with the improving edges. See Table 2 for a complete description (with respect to a bit configuration $\mathfrak{b}$). We say that a strategy σ is a phase p strategy with configuration $\mathfrak{b}$ iff every node is mapped by σ to a choice included in the respective cell of the table. Cells that contain more than one choice indicate that strategies of the respective phase are allowed to match any of the choices.

Table 2. Policy Phases (where $\mathfrak{b}' = \mathfrak{b} + 1$, $r = \nu^n_1(\mathfrak{b})$ and $r' = \nu^n_1(\mathfrak{b}')$)

Phase	1	2	3	4	5	6
$\bar{\sigma}(s)$	r	r	r	r	r	r'
$\bar{\sigma}(d^0_i)$	$1-\mathfrak{b}_{i+1}$	$1-\mathfrak{b}_{i+1}$	$1-\mathfrak{b}_{i+1}$	$1-\mathfrak{b}_{i+1}$	$1-\mathfrak{b}_{i+1}$	$1-\mathfrak{b}_{i+1}$, $1-\mathfrak{b}'_{i+1}$
$\bar{\sigma}(d^1_i)$	$\mathfrak{b}_{i+1}$	$\mathfrak{b}_{i+1}$	$\mathfrak{b}_{i+1}$	$\mathfrak{b}_{i+1}$	$\mathfrak{b}_{i+1}$	$\mathfrak{b}_{i+1}$, $\mathfrak{b}'_{i+1}$
$\bar{\sigma}(h^0_i)$	$\nu^n_{i+2}(\mathfrak{b})$	$\nu^n_{i+2}(\mathfrak{b}$	$\nu^n_{i+2}(\mathfrak{b})$	$\nu^n_{i+2}(\mathfrak{b})$, $\nu^n_{i+2}(\mathfrak{b}')$	$\nu^n_{i+2}(\mathfrak{b}')$	$\nu^n_{i+2}(\mathfrak{b}')$
$\bar{\sigma}(b^*_{*,*})$	0, r	0, r	0, r, r'	0, r'	0, r'	0, r'
$\bar{\sigma}(\mathrm{A}_i^{\mathfrak{b}_{i+1}})$	$\mathfrak{b}_i$	$*$	$*$	$*$	$*$	$*$
$\bar{\sigma}(\mathrm{A}_i^{\mathfrak{b}'_{i+1}})$	$*$	$\mathfrak{b}'_i$	$\mathfrak{b}'_i$	$\mathfrak{b}'_i$	$\mathfrak{b}'_i$	$\mathfrak{b}'_i$

Phase	1–2	3	4–6
$\bar{\sigma}(k_i)$	$\begin{cases} r & \text{if } \mathfrak{b}_i = 0 \\ -\mathfrak{b}_{i+1} & \text{if } \mathfrak{b}_i = 1 \end{cases}$	$\begin{cases} r, r' & \text{if } \mathfrak{b}'_i = 0 \text{ and } \mathfrak{b}_i = 0 \\ -\mathfrak{b}_{i+1}, r' & \text{if } \mathfrak{b}'_i = 0 \text{ and } \mathfrak{b}_i = 1 \\ -\mathfrak{b}'_{i+1} & \text{if } \mathfrak{b}'_i = 1 \end{cases}$	$\begin{cases} r' & \text{if } \mathfrak{b}'_i = 0 \\ -\mathfrak{b}'_{i+1} & \text{if } \mathfrak{b}'_i = 1 \end{cases}$

Phase 3	Side Conditions
(a)	$\forall i.\ \left(\left(\mathfrak{b}'_i = 0 \text{ and } (\exists j, l. \bar{\sigma}(b^j_{i,l}) = r')\right) \text{ implies } \bar{\sigma}(k_i) = r'\right)$
(b)	$\forall i, j.\ \left((\mathfrak{b}'_i = 0, \mathfrak{b}'_j = 0, \bar{\sigma}(k_i) = r' \text{ and } \bar{\sigma}(k_j) \neq r') \text{ implies } i > j\right)$

Table 3. Improving Switches (where $\mathfrak{b}' = \mathfrak{b} + 1$ and $r' = \nu_1^n(\mathfrak{b}')$)

Ph. p	Improving Switches Subset L_σ^p	Improving Switches Superset U_σ^p
1	$\{(b_{i,l}^j, \mathrm{A}_i^j) \mid \sigma(b_{i,l}^j) \neq \mathrm{A}_i^j\}$	L_σ^1
2	$\{(k_{r'}, c_{r'}^{\mathfrak{b}'_{r'+1}})\}$	$L_\sigma^1 \cup L_\sigma^2$
3	$\{(k_i, k_{r'}) \mid \bar{\sigma}(k_i) \neq r' \wedge \mathfrak{b}'_i = 0\} \cup$ $\{(b_{i,l}^j, k_{r'}) \mid \bar{\sigma}(b_{i,l}^j) \neq r' \wedge \mathfrak{b}'_i = 0\} \cup$ $\{(b_{i,l}^j, k_{r'}) \mid \bar{\sigma}(b_{i,l}^j) \neq r' \wedge \mathfrak{b}'_{i+1} \neq j\}$	$U_\sigma^4 \cup \{(k_i, k_z) \mid \bar{\sigma}(k_i) \notin \{z, r'\}, z \leq r' \wedge \mathfrak{b}'_i = 0\} \cup$ $\{(b_{i,l}^j, k_z) \mid \bar{\sigma}(b_{i,l}^j) \notin \{z, r'\}, z \leq r' \wedge \mathfrak{b}'_i = 0\} \cup$ $\{(b_{i,l}^j, k_z) \mid \bar{\sigma}(b_{i,l}^j) \notin \{z, r'\}, z \leq r' \wedge \mathfrak{b}'_{i+1} \neq j\}$
4	$\{(h_i^0, k_{\nu_{i+2}^n(\mathfrak{b}')}) \mid \bar{\sigma}(h_i^0) \neq \nu_{i+2}^n(\mathfrak{b}')\}$	$U_\sigma^5 \cup \{(h_i^0, k_l) \mid l \leq \nu_{i+2}^n(\mathfrak{b}')\}$
5	$\{(s, k_{r'})\}$	$U_\sigma^6 \cup \{(s, k_i) \mid \bar{\sigma}(s) \neq i \wedge i < r'\} \cup$ $\{(d_i^j, x) \mid \sigma(d_i^j) \neq x \wedge i < r'\}$
6	$\{(d_i^0, x) \mid \sigma(d_i^0) \neq x \wedge \bar{\sigma}(d_i^0) \neq \mathfrak{b}'_{i+1}\} \cup$ $\{(d_i^1, x) \mid \sigma(d_i^1) \neq x \wedge \bar{\sigma}(d_i^1) = \mathfrak{b}'_{i+1}\}$	$L_\sigma^1 \cup L_\sigma^6$

The following lemma tells us that all occurring values in the policy iteration are *small* compared to N^{2n+11}. Particularly, ε-times values are almost negligible.

Lemma 3. *Let σ be a policy belonging to one of the phases specified in Table 2. Then $|\mathrm{VAL}_\sigma(v)| < N^{2n+11}$ and $\varepsilon \cdot |\mathrm{VAL}_\sigma(v)| < 1$ for every node v.*

Table 3 specifies the sets of improving switches by providing for each phase p a subset L_σ^p and a superset U_σ^p s.t. $L_\sigma^p \subseteq I_\sigma \subseteq U_\sigma^p$. The intuition behind this method of giving the improving switches is that we will only *use* switches from L_σ^p while making sure that *no* other switches from U_σ^p are applied.

We finally arrive at the following main lemma describing the improving switches.

Lemma 4. *The improving switches from policies that belong to the phases in Table 2 are bounded by those specified in Table 3, i.e. $L_\sigma^p \subseteq I_\sigma \subseteq U_\sigma^p$ for a phase p policy σ.*

Note that phase 1 policies do not say anything about the particular configuration of inactive or open bicycles. To specify that all bicycles are either closed or completely opened, we say that a phase 1 policy σ is an *initial phase 1* policy if $\bar{\sigma}(b_{i,l}^j) = 0$ iff $\mathfrak{b}_i = 1$ and $\mathfrak{b}_{i+1} = j$.

Next, we specify the occurrence records w.r.t. $\mathfrak{b} \in \mathcal{B}_n$ that we want to have for an initial phase 1 policy σ. As described earlier, the entries of the occurrence records essentially depend on the number of bit flips of a certain index that have happened while counting up to $\mathfrak{b}$.

More precisely, we need to be able to count the number of occurred bit settings that match a certain *scheme*, which is a description of how a certain bit configuration should look like. Formally, a *scheme* is a set $S \subseteq (\mathbb{N} \setminus \{0\}) \times \{0, 1\}$. Let $\mathfrak{b} \in \mathcal{B}$. We write $S \models \mathfrak{b}$ iff $\mathfrak{b}_i = q$ for all $(i, q) \in S$. We can now define the set of bit configurations leading to $\mathfrak{b}$ that *match* the scheme. Formally, we define the *match set* as $M(\mathfrak{b}, S) = \{\mathfrak{b}' \leq \mathfrak{b} \mid S \models \mathfrak{b}'\}$.

We are most interested in schemes that correspond to flipping the i-th bit to one, i.e. schemes that demand for every bit $j < i$ to be zero. We define the *flip set* w.r.t. an index i and an additional scheme S by $F(\mathfrak{b}, i, S) = M(\mathfrak{b}, S \cup \{(i,1)\} \cup \{(j,0) \mid 0 < j < i\})$. We drop the parameter S if $S = \emptyset$.

We use the flip set to specify two numbers. First, we define the number of *bit flips* as the cardinality of the flip set by $f(\mathfrak{b}, i, S) = |F(\mathfrak{b}, i, S)|$. Second, we compute the *maximal flip number* representation in the flip set by $g(\mathfrak{b}, i, S) = \max(\{0\} \cup \{|\mathfrak{b}'| \mid \mathfrak{b}' \in F(\mathfrak{b}, i, S)\})$.

Table 4 specifies the occurrence record of an initial phase 1 policy. The technical conditions for the cycle components essentially say that (1) both cycle edges attached to A_i^j differ at most by one, that (2) the addition of both edges belonging to an active unset cycle equal $|\mathfrak{b}|$, that (3) the addition of both edges belonging to an active set cycle equal the maximal flip number when the respective bit was set, and that (4) recently opened inactive cycles are in the process of catching up with $|\mathfrak{b}|$ again.

Table 4. Occurrence Records

Edge e	$(*, t)$	(s, k_r)	(h_*^0, k_r)	$(b_{i,*}^j, k_r)$
$\phi^{\mathfrak{b}}(e)$	0	$f(\mathfrak{b}, r)$	$f(\mathfrak{b}, r)$	$f(\mathfrak{b}, r, \{(i,0)\}) + f(\mathfrak{b}, r, \{(i,1),(i{+}1, 1{-}j)\})$

Edge e	(k_i, k_r)	(k_i, c_i^j)	(d_i^j, s)	(d_i^j, h_i^j)
$\phi^{\mathfrak{b}}(e)$	$f(\mathfrak{b}, r, \{(i,0)\})$	$f(\mathfrak{b}, i, \{(i{+}1, j)\})$	$f(\mathfrak{b}, i{+}1) - j \cdot \mathfrak{b}_{i+1}$	$f(\mathfrak{b}, i{+}1) - (1{-}j) \cdot \mathfrak{b}_{i+1}$

Complicated Conditions

$$|\phi^{\mathfrak{b}}(b_{i,0}^j, \mathrm{A}_i^j) - \phi^{\mathfrak{b}}(b_{i,1}^j, \mathrm{A}_i^j)| \leq 1$$

$$\phi^{\mathfrak{b}}(b_{i,0}^j, \mathrm{A}_i^j) + \phi^{\mathfrak{b}}(b_{i,1}^j, \mathrm{A}_i^j) = \begin{cases} g^* + 1 & \text{if } \mathfrak{b}_i = 1 \text{ and } \mathfrak{b}_{i+1} = j \\ g^* + 1 + 2 \cdot z & \text{if } \mathfrak{b}_{i+1} \neq j \text{ and } z := |\mathfrak{b}| - g^* - 2^{i-1} < \frac{1}{2}(|\mathfrak{b}| - 1 - g^*), \\ |\mathfrak{b}| & \text{otherwise} \end{cases}$$

where $g^* = g(\mathfrak{b}, i, \{(i{+}1, j)\})$

We are now ready to specify our main lemma describing the transitioning from an initial phase 1 policy corresponding to $\mathfrak{b}$ to a successor initial phase 1 policy corresponding to $\mathfrak{b}'$, complying with the respective occurrence records.

Lemma 5. *Let σ be an initial phase 1 policy with configuration $\mathfrak{b} < \mathbf{1}_n$. There is an initial phase 1 policy σ' with configuration $\mathfrak{b}' = \mathfrak{b} + 1$ s.t. $(\sigma, \phi^{\mathfrak{b}}) \rightsquigarrow^+ (\sigma', \phi^{\mathfrak{b}'})$.*

It follows immediately that the MDPs provided here indeed simulate a binary counter.

Theorem 2. *The number of improving steps performed by* Least-Entered *on the MDPs constructed in this section, which contain $\mathcal{O}(n^2)$ vertices and edges, is $\Omega(2^n)$.*

The primal linear programs corresponding to the MDPs constructed in this section are thus linear programs on which the simplex algorithm with Zadeh's pivoting rule performs a subexponential number of iterations.

5 Concluding Remarks and Open Problems

We have shown that Zadeh's Least-Entered rule [25] may lead to a subexponential number of iterations by constructing explicit linear programs with n variables on which the expected number of iterations performed by Least-Entered is $2^{\Omega(\sqrt{n})}$.

The lower bound for linear programming has been obtained by constructing explicit parity games and subsequently MDPs on which we have the same expected number of iterations when solved by policy iteration. The lower bound result immediately transfers to mean payoff games, discounted payoff games and turned-based simple stochastic games [10].

The tie-breaking rule that we employed to prove the lower bound was non-explicit and definitely not a natural one. It would be interesting to see, whether it is easily possible to transform the MDPs presented here, in order to obtain an exponential lower bound for Zadeh's rule with a natural tie-breaking rule.

The most interesting open problems are, perhaps, whether linear programs can be solved in strongly polynomial time, whether the weak Hirsch conjecture holds, and whether there is a polynomial time algorithm for solving parity games or related game classes.

Acknowledgments. I would like to thank Uri Zwick and Thomas Dueholm Hansen for pointing me to this challenging pivoting rule and for numerous inspiring discussions on the subject. Also, I would like to thank the anonymous referees for their thorough reports that me helped to improve the presentation of this paper.

References

1. Avis, D., Chvátal, V.: Notes on Bland's pivoting rule. In: Polyhedral Combinatorics, Mathematical Programming Studies, vol. 8, pp. 24–34. Springer, Heidelberg (1978), http://dx.doi.org/10.1007/BFb0121192
2. Bertsekas, D.: Dynamic programming and optimal control, 2nd edn. Athena Scientific, Singapore (2001)
3. Bhat, G.S., Savage, C.D.: Balanced gray codes. Electronic Journal of Combinatorics 3, 2–5 (1996)
4. Broder, A., Dyer, M., Frieze, A., Raghavan, P., Upfal, E.: The worst-case running time of the random simplex algorithm is exponential in the height. Inf. Process. Lett. 56(2), 79–81 (1995)
5. Dantzig, G.: Linear programming and extensions. Princeton University Press, Princeton (1963)
6. Derman, C.: Finite state Markov decision processes. Academic Press, London (1972)
7. Fathi, Y., Tovey, C.: Affirmative action algorithms. Math. Program. 34(3), 292–301 (1986)
8. Fearnley, J.: Exponential lower bounds for policy iteration. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 551–562. Springer, Heidelberg (2010)

9. Friedmann, O.: An exponential lower bound for the parity game strategy improvement algorithm as we know it. In: Proc. of 24th LICS, pp. 145–156 (2009)
10. Friedmann, O.: An exponential lower bound for the latest deterministic strategy iteration algorithms. Selected Papers of the Conference "Logic in Computer Science 2009" (to appear) (2010), a preprint available from http://www.tcs.ifi.lmu.de/~friedman
11. Friedmann, O., Hansen, T., Zwick, U.: A subexponential lower bound for the random facet algorithm for parity games. In: Proc. of 22nd SODA (2011) (to appear)
12. Friedmann, O., Hansen, T., Zwick, U.: Subexponential lower bounds for randomized pivoting rules for solving linear programs (2011), a preprint available from http://www.tcs.ifi.lmu.de/~friedman
13. Gärtner, B., Henk, M., Ziegler, G.: Randomized simplex algorithms on Klee-Minty cubes. Combinatorica 18(3), 349–372 (1998), http://dx.doi.org/10.1007/PL00009827
14. Gärtner, B., Tschirschnitz, F., Welzl, E., Solymosi, J., Valtr, P.: One line and n points. Random Structures & Algorithms 23(4), 453–471 (2003), http://dx.doi.org/10.1002/rsa.10099
15. Goldfarb, D., Sit, W.: Worst case behavior of the steepest edge simplex method. Discrete Applied Mathematics 1(4), 277–285 (1979), http://www.sciencedirect.com/science/article/B6TYW-45GVXJ1-2B/2/a7035da2cf84d35e9503c69f883c23f7
16. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games. LNCS, vol. 2500. Springer, Heidelberg (2002)
17. Howard, R.: Dynamic programming and Markov processes. MIT Press, Cambridge (1960)
18. Jeroslow, R.G.: The simplex algorithm with the pivot rule of maximizing criterion improvement. Discrete Mathematics 4(4), 367–377 (1973), http://www.sciencedirect.com/science/article/B6V00-45FSNXP-1H/2/0968f0b25d2d8a2e0e160a8a248d06de
19. Kalai, G.: A subexponential randomized simplex algorithm (extended abstract). In: Proc. of 24th STOC. pp. 475–482 (1992)
20. Kalai, G.: Linear programming, the simplex algorithm and simple polytopes. Mathematical Programming 79, 217–233 (1997)
21. Klee, V., Minty, G.J.: How good is the simplex algorithm? In: Shisha, O. (ed.) Inequalities III, pp. 159–175. Academic Press, New York (1972)
22. Matoušek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. Algorithmica 16(4-5), 498–516 (1996)
23. Puterman, M.: Markov decision processes. Wiley, Chichester (1994)
24. Vöge, J., Jurdziński, M.: A discrete strategy improvement algorithm for solving parity games (Extended abstract). In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 202–215. Springer, Heidelberg (2000)
25. Zadeh, N.: What is the worst case behavior of the simplex algorithm? Tech. Rep. 27, Department of Operations Research, Stanford (1980)

An Iterative Scheme for Valid Polynomial Inequality Generation in Binary Polynomial Programming

Bissan Ghaddar[1,*], Juan C. Vera[2], and Miguel F. Anjos[3,**]

[1] Department of Management Sciences, University of Waterloo, Waterloo, ON, Canada, N2L 3G1
bghaddar@uwaterloo.ca
[2] Tilburg School of Economics and Management, Tilburg University, Tilburg, The Netherlands
j.c.veralizcano@uvt.nl
[3] Département de Mathématiques et génie industriel & GERAD, École Polytechnique de Montréal, Montréal, QC Canada H3T 1J4
anjos@stanfordalumni.org

Abstract. Semidefinite programming has been used successfully to build hierarchies of convex relaxations to approximate polynomial programs. This approach rapidly becomes computationally expensive and is often tractable only for problems of small sizes. We propose an iterative scheme that improves the semidefinite relaxations without incurring exponential growth in their size. The key ingredient is a dynamic scheme for generating valid polynomial inequalities for general polynomial programs. These valid inequalities are then used to construct better approximations of the original problem. As a result, the proposed scheme is in principle scalable to large general combinatorial optimization problems. For binary polynomial programs, we prove that the proposed scheme converges to the global optimal solution for interesting cases of the initial approximation of the problem. We also present examples illustrating the computational behaviour of the scheme and compare it to other methods in the literature.

Keywords: Binary Polynomial Programming, Binary Quadratic Programming, Valid Inequality Generation, Semidefinite Programming.

1 Introduction

Semidefinite programming is now well recognized as a powerful tool for combinatorial optimization. Early research in this vein has yielded improved approximation algorithms and very tight bounds for some hard combinatorial optimization

* Research supported by a Canada Graduate Scholarship from the Natural Sciences and Engineering Research Council of Canada.
** Research partially supported by the Natural Sciences and Engineering Research Council of Canada, and by a Humboldt Research Fellowship.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 207–222, 2011.

problems, see [22] and the references therein. As a consequence, interest in the application of semidefinite techniques to combinatorial optimization problems has continued unabated since the mid-1990s. This has included not only theoretical approximation guarantees but also the development and implementation of algorithms for efficiently solving semidefinite (SDP) relaxations of combinatorial optimization problems. Noteworthy developments in this direction include the biqmac solver for max-cut [35], an SDP-based branch-and-bound solver for max-k-cut [9], extremely tight bounds for the quadratic assignment problem [36], and exact solutions for single-row layout [1] as well as related quadratic linear ordering problems [11].

Since the seminal work of Lasserre [18], intense research activity has been carried out on polynomial programming (PP) and the related theory of moments. The main idea is the application of representation theorems to characterize the set of polynomials that are non-negative on a given domain. This research includes the recent work of de Klerk and Pasechnik [5], Lasserre [17,18], Laurent [20,21], Nie, Demmel, and Sturmfels [26], Parrilo [27,29], Peña, Vera, and Zuluaga [31,45], and the early work of Nesterov [24], Shor [39], and the $\mathcal{S}$-Lemma of Yakubovich (see [33]) among others. Specifically for binary optimization, the specialization of Lasserre's construction to binary PPs was shown to converge in a finite number of steps in [17] and the relationship of the Lasserre hierarchy to other well-known hierarchies was studied in works such as [19,20].

All the PP-based approaches rely on the construction of sums-of-squares (SOS) certificates of non-negativity for a suitable representation of the PP problem. While the resulting hierarchies yield very tight approximations of the original PP problem, from a computational perspective, these hierarchies all suffer from a common limitation, namely the explosion in size of the SDP relaxations involved. This fast growth in the size of the SDPs also affects formulations of combinatorial problems as PPs. One way to overcome this difficulty is to exploit the structure of the problem. This can be done either by taking advantage of symmetry as in Bai, de Klerk, Pasechnik, and Sotirov [2], de Klerk [4], de Klerk, Pasechnik, and Schrijver [6], de Klerk and Sotirov [7], and Gatermann and Parrilo [8], or by exploiting sparsity as in Kojima, Kim, and Waki [15], Waki, Kim, Kojima, and Muramatsu [41,42], and several others [12,13,14,16,25]. However, in the absence of any favorable structure, the practical application of the SOS approach is severely limited.

The motivation for our work is to devise ways to take advantage of the strength of the PP approach without depending on the presence of exploitable structure in the problem or paying a high computational price. To achieve this objective, it is imperative to avoid the growth of the complexity of the non-negativity certificates involved. For this purpose, instead of growing the degree of the certificates (which results in an exponential growth of the size of the relaxation), we fix the degree of the polynomials that we use and increase the set of polynomial inequalities describing the feasible set of the PP problem. These valid inequalities are then used to construct new certificates that provide better approximations. We obtain the valid inequalities by means of a dynamic inequality generation

scheme (DIGS) that makes use of information from the objective function to dynamically generate polynomial inequalities that are valid on the feasible region of the PP problem. The result is an iterative scheme that provides better and better SDP approximations without growing the degree of the certificates involved.

In this paper, we describe in some detail our proposed iterative scheme for general PP, and specialize it to binary PPs. Our method for the binary case can be seen as a generalization, from the linear case to higher degrees, of the lift and project methods of Balas, Ceria, and Cornuéjols [3], Sherali and Adams [37], and Lovász and Schrijver [23]. We prove that for binary PPs, the proposed scheme converges to the global optimal solution for interesting cases of the initial approximation of the problem. We also provide computational examples to highlight the advantages of the proposed scheme with respect to Lasserre's approach as well as to the lift-and-project method of Balas, Ceria, and Cornuéjols. The potential impact of the methodology presented here is significant, since it provides a means to tightly approximate binary PPs that, unlike previously proposed hierarchies of SDP relaxations, is in principle scalable to large general combinatorial optimization problems.

1.1 Polynomial Programming Problem

Consider the general PP problem whose objective and constraints are multivariate polynomials,

$$\begin{aligned} \textbf{(PP-P)}\ z = \ & \sup\ f(x) \\ & \text{s.t. } g_i(x) \geq 0 \qquad i = 1, \ldots, m. \end{aligned}$$

Let $S = \{x \in \mathbb{R}^n : g_i(x) \geq 0,\ i = 1, \ldots, m\}$, be the feasible set of (PP-P). We can rephrase (PP-P) as

$$\begin{aligned} \textbf{(PP-D)}\ & \inf\ \lambda \\ & \text{s.t. } \lambda - f(x) \geq 0 \quad \forall\, x \in S. \end{aligned} \tag{1}$$

The condition $\lambda - f(x) \geq 0$ for all $x \in S$ is $\mathcal{NP}$-hard for most (interesting) choices of S. To obtain computable relaxations of (PP-D), one uses tractable relaxations of condition (1) that can be re-phrased in terms of a linear system of equations involving positive semidefinite matrices [18,24,28,29,30,39,43], second-order cones [10], or linear problems [19,38,45].

1.2 Approximations Hierarchies for Polynomial Programs

Lasserre [18] introduced semidefinite relaxations corresponding to liftings of PPs into higher dimensions. The construction is motivated by results related to the representations of non-negative polynomials as SOS and the dual theory of moments. Lasserre shows that the global maximum of $f(x)$ over a compact set S defined by polynomial inequalities reduces to solving a sequence of SOS representations for polynomials that are non-negative on S. The convergence of

Lasserre's method is based on the assumption that $\{g_1(x), \ldots, g_m(x)\}$, the given description of S, allows the application of Putinar's Theorem [34]. In particular, it assumes that S is compact.

For ease of notation, define $g_0(x) = 1$ and $G = \{g_i(x) : i = 0, 1, \ldots, m\}$. For a given $r > 0$, a relaxation on the original polynomial program (PP-P) is obtained,

$$\begin{aligned} \mu_G^r = \inf_{\lambda, \sigma_i(x)} \ & \lambda \\ \text{s.t. } & \lambda - f(x) = \sum_{i=0}^{m} \sigma_0(x) g_i(x) \\ & \sigma_i(x) \text{ is SOS of degree} \leq (r - \deg(g_i)) \qquad i = 0, \ldots, m. \end{aligned} \tag{2}$$

Being an SOS of a given degree can be expressed in terms of SDP matrices, and thus the optimization problem (2) can be reformulated as a semidefinite program [39]. By increasing the value of r, one can build up a sequence of convex semidefinite relaxations of increasing size. Under mild conditions the optimal values of these problems converge to the global optimal value of the original non-convex problem (PP-P) [18,29].

Using Lasserre's approach for general polynomial programs, one may approach the global optimal value as closely as desired by solving a sequence of SDPs that grow in the size of the semidefinite matrices and in the number of constraints. The computational cost of the procedure clearly depends on both r and n, the number of variables. Problem (2) is computationally expensive in practice for $r > 2$ since the number of variables and constraints of (2) can be large, especially when using higher degree polynomials. For a problem with n variables and m inequality constraints, the optimization problem (2) has $m + 1$ semidefinite matrices of dimension $O(n^r)$ and $O(n^r)$ constraints.

1.3 Dynamic Approximations for Polynomial Programs

In this paper, we propose a scheme to dynamically generate valid polynomial inequalities for general PPs. Instead of growing r and increasing the size of the problem exponentially, we propose to fix r to a small value (mainly to d, the degree of $f(x)$) and improve the relaxation (2) by growing the set G, i.e., by adding valid polynomial inequalities to the description of S. Our approach makes use of information from the objective function to dynamically generate polynomial inequalities that are valid on the feasible region.

Before diving into the technical details, we use an example to show how the method works.

Example 1. Consider the non-convex quadratic knapsack problem with $n = 3$ and $d = 2$:

$$\begin{aligned} \max \ & 62x_1 + 19x_2 + 28x_3 + 52x_1x_2 + 74x_1x_3 + 16x_2x_3 \\ \text{s.t. } & 12x_1 + 44x_2 + 11x_3 \leq 66 \\ & x_1, x_2, x_3 \in \{0, 1\}. \end{aligned} \tag{3}$$

The optimal value for (3) is $z = 164$. Let $f(x) = 62x_1 + 19x_2 + 28x_3 + 52x_1x_2 + 74x_1x_3 + 16x_2x_3$. Setting $r = 2$ and replacing the condition $x_i \in \{0, 1\}$ with $0 \leq x_i \leq 1$ and $x_i^2 - x_i = 0$, we obtain the following relaxation

$$\begin{aligned} &\min \lambda \\ &\text{s.t. } \lambda - f(x) = s(x) + a(66 - 12x_1 - 44x_2 - 11x_3) + \sum_{i=1}^{3} b_i(1 - x_i) \\ &\qquad + \sum_{i=1}^{3} c_i x_i + \sum_{i=1}^{3} d_i(x_i - x_i^2), \\ &\quad s(x) \text{ is SOS of degree} \leq 2, a, b_i, c_i \in \mathbb{R}_+,\ d_i \in \mathbb{R} \end{aligned}$$

which has an objective value of 249.16. This is an upper bound on the optimal value of (3).

If one wants to improve the value using Lasserre's method, the hierarchy of SDPs shown in Table 1 must be solved.

Table 1. Results for Lasserre's hierarchy. The optimal solution is obtained with $r = 6$.

r	2	4	6	8
objective value	249.1	226.2	164.0	164.0
psd matrices	4×4(1)	10×10(1)	20×20(1)	35×35(1)
	1×1(13)	4×4(13),	10×10(13),	20×20(13)
total # of vars	26	135	925	3360
total # of constraints	10	35	84	165

By contrast, using our method we first generate a quadratic valid inequality

$$p(x) = 0.809 - 0.388x_1 - 0.037x_2 - 0.361x_3 - 0.099x_1x_2 - 0.086x_2x_3$$

and solve (2) with $r = 2$ again, this time taking $G = \{1, 66 - 12x_1 + 44x_2 + 11x_3, x_i, 1 - x_i, x_i^2 - x_i,$
$x_i - x_i^2, p(x)\}$. An objective function value of 243.22 is obtained. Performing this approach iteratively, one is able to improve the bound and obtain a tighter approximation of the original problem. After adding 11 inequalities we obtain an objective of 164.00 which is the optimal value of (3). In Table 2, the details on the size of the corresponding *master problem* and *polynomial generation subproblem* are given.

The key idea in the DIGS method is that if we can generate a new valid inequality for which we do not currently have a non-negativity certificate. We can use this new inequality, in combination with the previous ones, to generate new non-negativity certificates. This is formalized in Lemma 1. Before presenting Lemma 1 we define the notation we will use in the rest of the paper.

Table 2. Results for the proposed Dynamic Inequality Generation Scheme

i	0	1	2	3	$\cdots$	9	10	11
objective value	249.1	243.2	238.9	235.1	$\cdots$	164.2	164.0	164.0
Master Problem								
psd matrices	4x×4(1)	4×4(1)	4×4(1)	4×4(1)	$\cdots$	4×4(1)	4×4(1)	4×4(1)
non-negative vars	7	8	9	10	$\cdots$	16	17	18
free vars	3	3	3	3	$\cdots$	3	3	3
total # of vars	20	21	22	23	$\cdots$	29	30	31
total # of constraints	10	10	10	10	$\cdots$	10	10	10
Subproblem								
psd matrices	-	4×4(2)	4×4(2)	4×4(2)	$\cdots$	4×4(2)	4×4(2)	4×4(2)
non-negative vars	-	14	16	18	$\cdots$	30	32	34
free vars	-	20	20	20	$\cdots$	20	20	20
total # of vars	-	54	56	58	$\cdots$	70	72	74
total # of constraints	-	20	20	20	$\cdots$	20	20	20

Notation used in this paper: We use $\mathbf{R}[x] := \mathbf{R}[x_1, \ldots, x_n]$ (resp. $\mathbf{R}_d[x]$) to denote the set of polynomials in n variables with real coefficients (resp. of degree at most d). Given $S \subseteq \mathbb{R}^n$, we define $\mathcal{P}(S)$ (resp. $\mathcal{P}_d(S)$) to be the cone of polynomials (resp. of degree at most d) that are non-negative over S. We denote by Ψ (resp. Ψ_d) the cone of real polynomials (resp. of degree at most d) that are SOS of polynomials. Note $\Psi_d := \{\sum_{i=1}^{N} p_i(x)^2 : p(x) \in \mathbf{R}_{\lfloor \frac{d}{2} \rfloor}[x]\}$, with $N = \binom{n+d}{d}$, and in particular $\Psi_d = \Psi_{d-1}$ for every odd degree d.

For a given $r > 0$, define the approximation $\mathcal{K}_G^r$ of $\mathcal{P}_d(S)$ as:

$$\mathcal{K}_G^r = \left(\sum_{i=0}^{|G|} g_i(x) \Psi_{r - \deg(g_i)} \right) \cap \mathbf{R}_d[x].$$

Lemma 1. *Let $r \geq d$ and $p \in \mathcal{P}_d(S) \setminus \mathcal{K}_G^r$. Then*

$$\mathcal{K}_G^r \subsetneq \mathcal{K}_{G \cup \{p\}}^r \subseteq \mathcal{P}_d(S) \text{ and thus } \mu_G^r \geq \mu_{G \cup \{p\}}^r.$$

Thus, our inequality generation procedure is defined as

$$\texttt{GIVEN } G \texttt{ FIND } p(x) \in \mathcal{P}_d(S) \setminus \mathcal{K}_G^d. \tag{4}$$

In this procedure we need to tackle two problems: first, how to generate $p(x) \in \mathcal{P}_d(S)$ since somehow this is our original problem; and second, how to ensure $p(x) \notin \mathcal{K}_G^d$. This is the central point of the rest of the paper. We explain how DIGS works for general PP in Section 2, and present a specialized (and more efficient) DIGS for binary PP in Section 3. We show some convergence results for Binary Quadratic Programming in Section 3.2. We also give examples comparing our method to Lasserre's method and to the Balas, Ceria, and Cornuéjols lift-and-project in Section 3.3.

2 General Case

2.1 Dynamic Inequality Generation Scheme (DIGS)

To execute Procedure (4):

$$\texttt{GIVEN } G \texttt{ FIND } p(x) \in \mathcal{P}_d(S) \setminus \mathcal{K}_G^d$$

We will generate only polynomials for which we can ensure non-negativity on S. To do this we will generate polynomials in $\mathcal{K}_G^{d+2}$. To ensure $p(x) \notin \mathcal{K}_G^d$, we use the dual optimal solution of (2), denoted by Y. We will abuse the notation and we identify $\mathbf{R}_d[x]$ with $\mathbb{R}^N$ where $N = \binom{n+d}{d}$, e.g. by identifying each polynomial $f(x) \in \mathbf{R}_d[x]$ with its vector of coefficients $f \in \mathbb{R}^N$. In this way $\mathcal{K}_G^d$ is a cone in $\mathbb{R}^N$. We endow $\mathbb{R}^N$ with the inner product $\langle \cdot, \cdot \rangle$ such that for each $f(x) \in \mathbf{R}_d[x]$ and each $u \in \mathbb{R}^n$, $\langle f, \mathcal{M}_d(u) \rangle = f(u)$, where $\mathcal{M}_d(u)$ is the vector of monomials of u up to degree d. In this way, relaxation (2) and its SDP dual correspond to the conic primal-dual pair:

$$\begin{array}{ll} \inf_\lambda \ \lambda & \sup_Y \ \langle f, Y \rangle \\ \text{s.t.} \ \ \lambda - f(x) \in \mathcal{K}_G^d & \text{s.t.} \ \ \langle 1, Y \rangle = 1 \\ & \qquad Y \in (\mathcal{K}_G^d)^*. \end{array} \tag{5}$$

From the definition of dual cone $(\mathcal{K}_G^d)^*$, we have the following lemma,

Lemma 2. *Let Y be a solution of (5). For all $p \in \mathcal{K}_G^d$, $\langle p, Y \rangle \geq 0$.*

Thus to generate $p(x) \in \mathcal{K}_G^{d+2} \setminus \mathcal{K}_G^d$, one can solve the following SDP problem. We refer to this problem as the polynomial generating subproblem:

$$\begin{aligned} \textbf{(PP-Sub)} \ \ \min \ & \langle p, Y \rangle \\ \text{s.t.} \ & p(x) \in \mathcal{K}_G^{d+2} \\ & \| \, p \, \| \leq 1. \end{aligned} \tag{6}$$

The normalization constraint is added since $p(x)$ and $cp(x)$ are equivalent inequalities for any $c > 0$. Moreover, without the normalization constraint, (PP-Sub) is unbounded. There are several options for choosing the norm $\| \cdot \|$. In this paper we use the one that maximizes the ℓ_2 distance between Y and the set $\{\mathcal{M}_d(x) : p(x) = 0\}$.

Example 2.

$$\begin{aligned} \min \quad & x_1 - x_1x_3 - x_1x_4 + x_2x_4 + x_5 - x_5x_7 - x_5x_8 + x_6x_8 \\ \text{s.t.} \ & x_3 + x_4 \leq 1 \\ & x_7 + x_8 \leq 1 \\ & 0 \leq x_i \leq 1 \qquad \forall i \in \{1, \ldots, 8\}. \end{aligned}$$

The optimal objective value of the above problem is 0. We need to go at least up to $r = 10$ for Lasserre's hierarchy to obtain the optimal value. However, for

$r = 8$ even constructing the problem couldn't be done within an hour. Using the dynamic scheme, we are able to use relaxations of degree 2 and add inequalities as in the previous example. In this case we stop after 50 inequalities due to the slow improvement in the bound, obtaining a lower bound of value -0.014 in 200.1 seconds. Figure 1 illustrates the bound improvement.

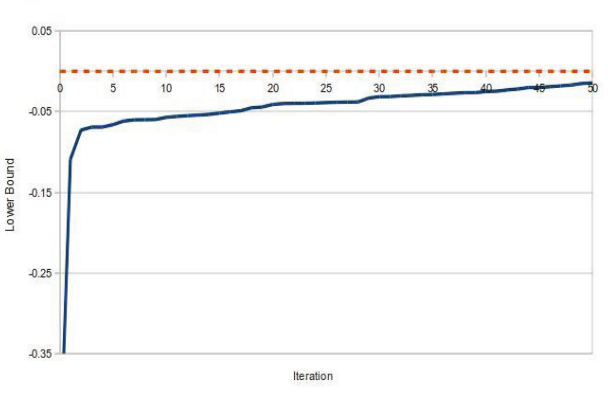


Fig. 1. DIGS lower bounds. The dotted line is the optimal objective value.

3 Binary Case

In this section we specialize the results presented in Section 2 to PPs where (some) all of the variables are binary; we refer to such problems as (mixed) binary polynomial programs (BPP). Algebraic geometry representation techniques are used to obtain a scheme that iteratively improves the bound and converges to the optimal objective value of the original binary polynomial program. Using the approach proposed in [31] and [44], we obtain a computationally cheaper subproblem for the binary case. Further, we present convergence results for some important cases. We prove that the resulting iterative scheme converges to the global optimal solution of the binary polynomial program when starting from the exact representation of the domain set excluding the binary constraints. As such a representation is not tractable in general, we show an initial approximation ensuring convergence for binary polynomial programs with a quadratic objective function and a set of linear constraints.

3.1 Specializing the Dynamic Inequality Generation Scheme

Using (PP-D), BPP can be reformulated as

$$\begin{aligned} z \;=\; & \inf \lambda \\ & \text{s.t. } \lambda - f(x) \in \mathcal{P}_d(D \cap \{-1,1\}^n), \end{aligned} \tag{7}$$

where $D = \{x : g_i(x) \geq 0,\ i = 1, \ldots, m\}$. Without loss of generality we assume $D \subseteq [-1,1]^n$. Let $H_j = \{x \in \mathbb{R}^n : x_j \in \{-1,1\}\}$ and $H = \{-1,1\}^n$. Notice that $H = \cap_{j=1}^n H_j$.

To solve (7), we follow an approach similar to that for the general case presented in Section 2. Let $G = \{g_0(x), g_1(x), \ldots, g_m(x)\}$ where $g_0(x) = 1$. Define $\mathcal{Q}_G^r$ as the following approximation to $\mathcal{P}_d(D \cap H)$:

$$\mathcal{Q}_G^r = \left(\sum_{i=0}^{m} g_i(x) \Psi_{r-\deg(g_i)} + \sum_{i=1}^{n} (1 - x_i^2) \mathbf{R}_{r-2}[x] \right) \cap \mathbf{R}_d[x]$$

and define the PP master problem

$$\begin{aligned} \varphi \ &= \inf_{\lambda} \lambda \\ &\text{s.t. } \lambda - f(x) \in \mathcal{Q}_G^d, \end{aligned} \tag{8}$$

which can be reformulated as an SDP. In this setting, for the polynomial generating subproblem (6), instead of solving the subproblem over $\mathcal{Q}_G^{d+2}$ as defined in Section 2.1, we use the following theorem to obtain a computationally cheaper algorithm.

Theorem 1. *[31] For any degree d and compact set D,*

$$\mathcal{P}_d(D \cap H_j) = ((1 + x_j)\mathcal{P}_d(D) + (1 - x_j)\mathcal{P}_d(D) + (1 - x_j^2)\mathbf{R}_{d-1}[x]) \cap \mathbf{R}_d[x].$$

From Theorem 1, it is natural to define the operator

$$\mathcal{C}_j^d(\mathcal{Q}) := \big((1 + x_j)\mathcal{Q} + (1 - x_j)\mathcal{Q} + (1 - x_j^2)\mathbf{R}_{d-1}[x]\big) \cap \mathbf{R}_d[x],$$

for any $\mathcal{Q} \subseteq \mathbf{R}_d[x]$. The following lemma is the key to our DIGS for the binary case.

Lemma 3. *Assume $S = D \cap H$, and let $\mathcal{Q} \subseteq \mathcal{P}_d(S)$.*

1. *For every j, $\mathcal{Q} \subseteq \mathcal{C}_j^d(\mathcal{Q}) \subseteq \mathcal{P}_d(S)$.*
2. *Moreover, if $\mathcal{P}_d(D) \subseteq \mathcal{Q}$ then $\mathcal{P}_d(D \cap H_j) \subseteq \mathcal{C}_j^d(\mathcal{Q})$.*
3. *If $\mathcal{P}_d(D) \subseteq \mathcal{Q} \subsetneq \mathcal{P}_d(S)$ then for some j, $\mathcal{Q} \subsetneq \mathcal{C}_j^d(\mathcal{Q})$.*

Let Y be the dual optimal solution for (8). Define the j-th valid inequality generating subproblem as:

$$\begin{aligned} \omega_j \ &= \ \min \ \langle p, Y \rangle \\ &\text{s.t. } p(x) \in \mathcal{C}_j^d(\mathcal{Q}_G^d) \\ &\quad \| \, p \, \| \leq 1. \end{aligned} \tag{9}$$

Using the master problem (8) and the subproblem (9) iteratively, we obtain a DIGS for the binary case. The algorithm terminates when for all indexes j, the subproblems have value equal to zero. In practice, we stop when for all j the value of the subproblem is sufficiently close to zero.

The DIGS obtained for the binary case is computationally more efficient than the DIGS presented in Section 2.1 for the general case. The master problem in

both cases is of the same size, but solving the subproblem (9) is basically of the same order as solving the master problem (8). Subproblem (9) has twice the number of variables and $\frac{n+d+1}{d+1}$ times the number of constraints of the master problem. This is much smaller than subproblem (6) obtained for the general case which has $O(n^2/d^2)$ times the number of variables and $O(n^2/d^2)$ times the number of constraints compared to the master problem.

3.2 Convergence Results

In this section, we first start by providing a proof of convergence for the case when the approximation $\mathcal{Q}_G^d$ contains $\mathcal{P}_d(D)$ in the master problem. We use this theorem to show convergence of the DIGS for unconstrained binary polynomial programs with quadratic objective function, and for binary polynomial programs with quadratic objective function and linear constraints.

Notice that for the case where $\mathcal{Q}_G^d$ is replaced with $\mathcal{P}_d(D)$ in the master problem (8), the subproblem (9) is equivalent to optimizing over $\mathcal{C}_j^d(\mathcal{P}_d(D)) = \mathcal{P}_d(D \cap H_j)$, for a given index j. So intuitively, if the subproblem has a value of 0, it is because using the fact that x_j is binary cannot help, i.e., the solution is already "binary" in that coordinate. Thus if the value of all subproblems is zero we have converged to the optimal value. This intuition is formally expressed in Theorem 2. Recall from (5) that the dual optimal solution $Y \in \{X \in \mathcal{P}_d(D)^* : \langle 1, X\rangle = 1\}$.

Lemma 4. *Let $D \subseteq \mathbb{R}^n$ be a compact set, then*

$$\{X \in \mathcal{P}_d(D)^* : \langle 1, X\rangle = 1\} = conv(\mathcal{M}_d(D)).$$

Theorem 2. *Assume $\mathcal{P}_d(D) \subseteq \mathcal{Q}_G^d \subseteq \mathcal{P}_d(S)$. Let ω_j, φ, z be the optimal objective value of subproblem (9), master problem (8), and the original binary polynomial program (7) respectively. Assume $D \subseteq [-1, 1]^n$ and $d \geq 2$. If $\omega_j = 0$ for all indexes j, then $\varphi = z$.*

Proof. Let (λ, Y) be the optimal primal-dual solution of (8). Then $\varphi = \lambda \geq z$ and $Y \in \{X \in (\mathcal{Q}_G^d)^* : \langle 1, X\rangle = 1\} \subseteq \text{conv}(\mathcal{M}_d(D))$ using Lemma 4. D is a compact set. By Caratheodory's Theorem, Y can be written as $Y = \sum_i a_i \mathcal{M}_d(u_i)$ with $a_i > 0$, $\sum_i a_i = 1$ and each $u_i \in D$.
Notice that $\langle f, Y\rangle = \sum_i a_i \langle f, \mathcal{M}_d(u_i)\rangle = \sum_i a_i f(u_i)$. If $u_i \in H$ for all i, $\varphi = \langle f, Y\rangle \leq z$ and we are done. To get a contradiction, assume $u_k \notin H$ for some k. Then there is $j \leq n$ such that $u_k \notin H_j$. Consider $p(x) = 1 - x_j^2$. We have $p \in \mathcal{P}_d(H_j) \subseteq \mathcal{P}_d(D \cap H_j) \subseteq \mathcal{Q}_G^d$, and $p(u_k) > 0$. Therefore,

$$\omega_j \geq \langle p, Y\rangle = \sum_i a_i \langle p, \mathcal{M}_d(u_i)\rangle = \sum_i a_i p(u_i) \geq a_k p(u_k) > 0,$$

which is a contradiction. □

In the case of pure quadratic binary programming, taking D as the ball $\mathcal{B} = \{x \in \mathbb{R}^n : \|x\|^2 \leq n\}$ and $d = 2$, it follows from the $\mathcal{S}$-lemma that $\mathcal{P}_2(D) = \Psi_2 + (n - \|x\|^2)\mathbf{R}_0^+[x]$ and thus $\mathcal{P}_d(D) \subseteq \mathcal{Q}_G^d \subseteq \mathcal{P}_d(S)$ holds. From Theorem 2, we obtain convergence in the case of unconstrained binary quadratic programming.

Theorem 3. *When DIGS is applied to the case of pure binary quadratic programming, starting with* $G_0 = \{n - \|x\|^2, 1\}$*, if all the subproblems have optimal value 0, then the objective function value of the master problem is equal to the optimal value of the original binary problem.*

In general, for $S = \{x : A^T x = b\}$, the decision problem for $\mathcal{P}_2(S)$ is $\mathcal{NP} - hard$, and thus we do not have an efficient description for $\mathcal{P}_2(S)$ unless $\mathcal{P} = \mathcal{NP}$. Thus we can not apply directly Theorem 2. However, a modification of the proof of Theorem 2 shows that if G contains $(a_i^T x - b_i)^2$ for each linear equality $a_i^T x = b_i$, then the u's constructed in the proof would be in S.

Theorem 4. *When DIGS is applied to a binary quadratic programming problem constrained to* $A^T x = b$*, starting with* $G_0 = \{1, n - \|x\|^2, (a_i^T x - b_i)^2, -(a_i^T x - b_i)^2\}$*, if all the subproblems have an optimal value of 0, then the optimal value of the master problem is equal to the optimal value of the original binary problem.*

3.3 Examples

In this section, we provide several examples of BPPs and present computational results for DIGS. As our first two examples we consider the quadratic knapsack problem and the quadratic assignment problem. We report the objective function value at iteration zero and after performing 1, 5, and 10 iterations of DIGS. To solve these examples, we developed a MATLAB code that constructs and builds the resulting relaxations of the polynomial program and solves them using the SeDuMi solver [40]. As a reference for comparison we also present Lasserre's results reporting the order of the relaxation to get the global optimal solution and the corresponding objective function value. In case where the time limit of one hour is reached, we report the bounds for Lasserre's relaxation and the highest order r that could be solved within the time limit. To obtain a fair comparison, Lasserre's relaxation was solved using the same code, on the same machine.

Example 3. **Quadratic Knapsack Problem.** Given n items with a non-negative weight w_i assigned to each item i, and a profit matrix P, the quadratic knapsack problem maximizes the profit subject to a capacity constraint:

$$\begin{aligned} \max\ & x^T P x \\ \text{s.t. } & w^T x \leq c \\ & x \in \{0,1\}^n. \end{aligned}$$

Table 3 presents computational results for small quadratic knapsack instances where the parameters are generated according to [32]. The results show that DIGS is much more efficient, in particular when n gets large. For $n = 20$, we are not able to go beyond $r = 2$ for Lasserre's hierarchy in the given time limit.

Example 4. **Quadratic Assignment Problem.** Consider the quadratic assignment problem where we want to allocate a set of n facilities to a set of n locations,

Table 3. Computational results for quadratic knapsack instances. Values in bold are optimal.

		Lasserre			DIGS				
n	Optimal	r	obj.	t(sec)	Iter. 0	Iter. 1	Iter. 5	Iter. 10	t(sec)
5	370	4	**370.0**	2.1	413.9	399.4	**370.0**	-	6.0
10	1679	4	1707.3	28.1	1857.7	1821.9	1796.9	1791.5	7.2
15	2022	4	**2022.0**	1150.8	2270.5	2226.8	2180.4	2150.1	18.1
20	8510	2	9060.3	2.9	9060.3	9015.3	8925.9	8850.3	35.4
30	18229	2	19035.9	4.3	19035.9	18920.2	18791.7	18727.2	196.6

with the cost being a function of the distance d and flow between the facilities f. The objective is to assign each facility to a location such that the total cost is minimized. This can be formulated as:

$$\begin{aligned} &\min \textstyle\sum_{i\neq k, j\neq l} f_{ik} d_{jl} x_{ij} x_{kl} \\ &\text{s.t.} \textstyle\sum_i x_{ij} = 1 && 1 \leq j \leq n \\ &\quad \textstyle\sum_j x_{ij} = 1 && 1 \leq i \leq n \\ &\quad x \in \{0,1\}^{n\times n}. \end{aligned}$$

Table 4. Computational results for quadratic assignment instances. Values in bold are optimal.

		Lasserre			DIGS				
n	Optimal	r	obj.	t(sec)	Iter. 0	Iter. 1	Iter. 5	Iter. 10	t(sec)
3	46	2	**46.0**	0.3	**46.0**	-	-	-	0.3
4	56	2	50.8	1.0	50.8	51.8	**52.0**	-	6.3
5	110	2	104.3	3.4	104.3	105.1	106.3	106.8	68.5
6	272	2	268.9	9.3	268.9	269.4	269.8	270.2	404.4

Table 4 presents computational results for small quadratic assignment instances where f_{ik} and d_{jl} are integers generated uniformly between 0 and 5. Using Lasserre's hierarchy we can only go up to $r = 2$ for instances of dimension $n \geq 4$ within one hour, while the dynamic scheme after 10 iterations improves significantly on the bounds of Lasserre's $r = 2$ relaxation without as much computational effort.

As a final example, we consider the maximum stable set problem. In this case, we compare DIGS with the lift-and-project method of Balas et al. [3]. This comparison provides a fair indication of the advantages of our method in terms of bound quality. For each instance we impose a 300 seconds time limit for each procedure. The upper bound for lift-and-project is compared to three approaches of DIGS. Linear refers to generating linear inequalities that are added to the master problem by using a non-negative multiplier. SOCP refers to generating linear inequalities that are added to the master problem by using a polynomial multiplier that is in $\mathcal{P}_1(\mathcal{B})$ [10]. Quadratic refers to generating quadratic

inequalities similar to the previous examples described. Since our methodology is a generalization of the lift-and-project method, our algorithm was used to obtain the Balas et al. results for the maximum stable set.

Example 5. **Stable Set Problem.** Consider an undirected graph $G(V, E)$ where V and E are the vertex set and the edge set respectively. Given a subset $\bar{V} \subseteq V$, then $\bar{V}$ is called a stable set of G if there is no edge connecting any two vertices in $\bar{V}$. The maximum stable set problem is to find a stable set of maximal cardinality. Letting $n = |V|$, the maximum stable set problem can be formulated as a binary problem as follows:

$$\begin{aligned} \textbf{(SS-D2)} \quad & \max \textstyle\sum_i x_i \\ & \text{s.t. } x_i x_j = 0 \qquad \forall (i,j) \in E \\ & \quad x \in \{0,1\}^n. \end{aligned}$$

It can also be formulated as a linear problem by replacing the constraint $x_i x_j = 0$ by $x_i + x_j \leq 1$, we refer to this problem as (SS-LP).

Table 5. Computational results for the stable set problem with a time limit of 300 seconds

		(SS-LP)	Balas et al.		(SS-D2)	Linear		SOCP		Quadratic	
n	Optimal	UB	UB	Iter.	UB	UB	Iter.	UB	Iter.	UB	Iter.
8	3	4.00	3.00	197	3.44	3.00	186	3.00	126	3.02	49
11	4	5.50	4.00	160	4.63	4.00	139	4.00	130	4.05	109
14	5	7.00	5.02	135	5.82	5.02	114	5.01	91	5.14	82
17	6	8.50	6.22	121	7.00	6.23	84	6.09	63	6.30	54
20	7	10.00	7.46	104	8.18	7.43	68	7.25	45	7.42	38
23	8	11.50	8.81	88	9.36	8.61	50	8.36	33	8.67	22
26	9	13.00	10.11	77	10.54	9.84	37	9.60	25	9.96	14
29	10	14.50	11.65	65	11.71	11.10	24	10.87	17	11.18	10
32	11	16.00	13.03	56	12.89	12.37	18	12.20	14	12.53	6
35	12	17.50	14.48	49	14.07	13.49	13	13.32	10	13.66	4
38	13	19.00	16.05	43	15.24	14.80	8	14.74	7	14.85	4
41	14	20.50	17.69	39	16.42	15.88	7	15.77	6	16.26	1
44	15	22.00	19.10	34	17.59	17.19	6	17.09	5	17.30	1
47	16	23.50	20.78	29	18.77	18.39	4	18.26	4	18.59	1
50	17	25.00	22.18	27	19.94	19.52	4	19.42	4	19.77	1

Table 5 shows the upper bounds and the number of iterations performed within a time limit of 300 seconds. Lift-and-project performs the largest number of iterations for these instances since it utilizes linear programming which is computationally more efficient, however this efficiency comes at the expense of the bounds. For all instances the bounds obtained by DIGS using SOCP type of inequalities are the best bounds obtained within 300 seconds. These bounds are comparable with those from the Linear and Quadratic approaches, however Quadratic performs the least number of iterations and still achieves a competitive bound.

References

1. Anjos, M.F., Vannelli, A.: Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes. INFORMS J. Comp. 20(4), 611–617 (2008)
2. Bai, Y., de Klerk, E., Pasechnik, D., Sotirov, R.: Exploiting group symmetry in truss topology optimization. Optimization and Engineering 10(3), 331–349 (2009)
3. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. Mathematical Programming 58, 295–324 (1993)
4. de Klerk, E.: Exploiting special structure in semidefinite programming: A survey of theory and applications. European Journal of Operational Research 201(1), 1–10 (2010)
5. de Klerk, E., Pasechnik, D.: Approximation of the stability number of a graph via copositive programming. SIAM Journal on Optimization 12(4), 875–892 (2002)
6. de Klerk, E., Pasechnik, D., Schrijver, A.: Reduction of symmetric semidefinite programs using the regular*-representation. Mathematical Programming 109(2-3), 613–624 (2007)
7. de Klerk, E., Sotirov, R.: Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. Mathematical Programming 122(2), 225–246 (2010)
8. Gatermann, K., Parrilo, P.: Symmetry groups, semidefinite programs, and sums of squares. Journal of Pure and Applied Algebra 192(1-3), 95–128 (2004)
9. Ghaddar, B., Anjos, M.F., Liers, F.: A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem. Annals of Operations Research (to appear)
10. Ghaddar, B., Vera, J., Anjos, M.F.: Second-order cone relaxations for binary quadratic polynomial programs. SIAM Journal on Optimization (to appear)
11. Hungerländer, P., Rendl, F.: Semidefinite relaxations of ordering problems. Technical report, Alpen-Adria-Universität Klagenfurt (August. 2010)
12. Kim, S., Kojima, M., Mevissen, M., Yamashita, M.: Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion. To appear in Mathematical Programming (2009)
13. Kim, S., Kojima, M., Toint, P.: Recognizing underlying sparsity in optimization. Mathematical Programming 9(2), 273–303 (2009)
14. Kobayashi, K., Kim, S., Kojima, M.: Correlative sparsity in primal-dual interior-point methods for LP, SDP and SOCP. Applied Mathematics and Optimization 58(1), 69–88 (2008)
15. Kojima, M., Kim, S., Waki, H.: Sparsity in sums of squares of polynomials. Mathematical Programming 103(1), 45–62 (2003)
16. Kojima, M., Muramatsu, M.: A note on sparse SOS and SDP relaxations for polynomial optimization problems over symmetric cones. Computational Optimization and Applications 42(1), 31–41 (2009)
17. Lasserre, J.: An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. SIAM Journal on Optimization 12(3), 756–769 (2001)
18. Lasserre, J.: Global optimization problems with polynomials and the problem of moments. SIAM Journal on Optimization 11, 796–817 (2001)
19. Lasserre, J.: Semidefinite programming vs. LP relaxations for polynomial programming. Mathematics of Operations Research 27(2), 347–360 (2002)
20. Laurent, M.: A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0-1 programming. Mathematics of Operations Research 28, 470–496 (2001)

21. Laurent, M.: Semidefinite representations for finite varieties. Mathematical Programming 109(Ser. A), 1–26 (2007)
22. Laurent, M., Rendl, F.: Semidefinite programming and integer programming. In: Handbook on Discrete Optimization, vol. 12, pp. 393–514. Elsevier, Amsterdam (2005)
23. Lovász, L., Schrijver, A.: Cones of matrices and set-functions and 0-1 optimization. SIAM Journal on Optimization 1, 166–190 (1991)
24. Nesterov, Y.: Structure of non-negative polynomials and optimization problems. Technical report, Technical Report 9749, CORE (1997)
25. Nie, J., Demmel, J.: Sparse SOS relaxations for minimizing functions that are summation of small polynomials. SIAM Journal on Optimization 19(4), 1534–1558 (2008)
26. Nie, J., Demmel, J., Sturmfels, B.: Minimizing polynomials via sum of squares over the gradient ideal. Mathematical Programming: Series A and B 106(3), 587–606 (2006)
27. Parrilo, P.: Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. PhD thesis, Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, California (2000)
28. Parrilo, P.: An explicit construction of distinguished representations of polynomials nonnegative over finite sets. Technical report, IFA Technical Report AUT02-02, Zurich - Switzerland (2002)
29. Parrilo, P.: Semidefinite programming relaxations for semialgebraic problems. Mathematical Programming 96(2), 293–320 (2003)
30. Parrilo, P., Sturmfels, B.: Minimizing polynomial functions, algorithmic and quantitative real algebraic geometry. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 60, 83–89 (2003)
31. Peña, J.F., Vera, J.C., Zuluaga, L.F.: Exploiting equalities in polynomial programming. Operations Research Letters 36(2) (2008)
32. Pisinger, D.: The quadratic knapsack problem-a survey. Discrete Applied Mathematics 155(5), 623–648 (2007)
33. Pólik, I., Terlaky, T.: A survey of the $\mathcal{S}$-lemma. SIAM Review 49, 371–418 (2007)
34. Putinar, M.: Positive polynomials on compact semi-algebraic sets. Indiana University Mathematics Journal 42, 969–984 (1993)
35. Rendl, F., Rinaldi, G., Wiegele, A.: A branch and bound algorithm for max-cut based on combining semidefinite and polyhedral relaxations. Integer programming and combinatorial optimization 4513, 295–309 (2007)
36. Rendl, F., Sotirov, R.: Bounds for the quadratic assignment problem using bundle method. Mathematical Programming, Series B 109, 505–524 (2007)
37. Sherali, H.D., Adams, W.P.: A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM Journal on Discrete Mathematics 3(3), 411–430 (1990)
38. Sherali, H.D., Tuncbilek, C.H.: Comparison of two reformulation-linearization technique based linear programming relaxations for polynomial programming problems. Journal of Global Optimization 10(4), 381–390 (1997)
39. Shor, N.: A class of global minimum bounds of polynomial functions. Cybernetics 23(6), 731–734 (1987)
40. Sturm, J.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optimization Methods and Software, 11–12 (1999)
41. Waki, H., Kim, S., Kojima, M., Muramatsu, M.: Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity. SIAM Journal on Optimization 17(1), 218–242 (2006)

42. Waki, H., Kim, S., Kojima, M., Muramatsu, M.: SparsePOP: a sparse semidefinite programming relaxation of polynomial optimization problems. ACM Transactions on Mathematical Software 35(2), 15 (2008)
43. Wolkowicz, H., Saigal, R., Vandenberghe, L.: Handbook of Semidefinite programming -Theory, Algorithms, and Applications. Kluwer, Dordrecht (2000)
44. Zuluaga, L.: A conic programming approach to polynomial optimization problems: Theory and applications. PhD thesis, The Tepper School of Business, Carnegie Mellon University, Pittsburgh (2004)
45. Zuluaga, L., Vera, J.C., Peña, J.: LMI approximations for cones of positive semidefinite forms. SIAM Journal on Optimization 16(4) (2006)

A New Approach to the Stable Set Problem Based on Ellipsoids

Monia Giandomenico[1], Adam N. Letchford[2], Fabrizio Rossi[1], and Stefano Smriglio[1]

[1] Department of Computer Science, University of L'Aquila, Italy
{giandomenico,rossi,smriglio}@di.univaq.it
[2] Department of Management Science, Lancaster University, United Kingdom
A.N.Letchford@lancaster.ac.uk

Abstract. A new exact approach to the stable set problem is presented, which attempts to avoids the pitfalls of existing approaches based on linear and semidefinite programming. The method begins by constructing an ellipsoid that contains the stable set polytope and has the property that the upper bound obtained by optimising over it is equal to the Lovász theta number. This ellipsoid is then used to derive cutting planes, which can be used within a linear programming-based branch-and-cut algorithm. Preliminary computational results indicate that the cutting planes are strong and easy to generate.

Keywords: stable set problem, semidefinite programming, convex quadratic programming, cutting planes.

1 Introduction

Given an undirected graph $G = (V, E)$, a *stable set* is a set of pairwise non-adjacent vertices. The *stable set problem* (SSP) calls for a stable set of maximum cardinality. The SSP is $\mathcal{NP}$-hard [15], hard to approximate [14], and hard to solve in practice (e.g., [7, 23–25]). Moreover, it is a remarkable fact that sophisticated mathematical programming algorithms for the SSP, such as those in [4, 6, 12, 23, 25], have not performed significantly better than relatively simple algorithms based on implicit enumeration, such as those in [7, 24].

A possible explanation for the failure of mathematical programming approaches is the following. Linear Programming (LP) relaxations can be solved reasonably quickly, but tend to yield weak upper bounds. Semidefinite Programming (SDP) relaxations, on the other hand, typically yield much stronger bounds, but take longer to solve. Therefore, branch-and-bound algorithms based on either LP or SDP relaxations are slow, due to the large number of nodes in the search tree, or the long time taken to process each node, respectively.

In this paper we present a way out of this impasse. The key concept is that one can efficiently construct an *ellipsoid* that contains the stable set polytope, in such a way that the upper bound obtained by optimising over the ellipsoid

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 223–234, 2011.

is equal to the standard SDP bound, the so-called *Lovász theta number*. This ellipsoid can then be used to construct useful convex programming relaxations of the stable set problem or, more interestingly, to derive cutting planes. These cutting planes turn out to be strong and easy to generate.

We remark that our approach can be applied to the variant of the SSP in which vertices are weighted, and one seeks a stable set of maximum weight.

The paper is structured as follows. Some relevant literature is reviewed in Sect. 2, the new approach is presented in Sect. 3, some computational results are given in Sect. 4, and some concluding remarks are made in Sect. 5.

2 Literature Review

We now review the relevant literature. From this point on, $n = |V|$ and $m = |E|$.

2.1 Linear Programming Relaxations

The SSP has the following natural formulation as a 0-1 LP:

$$\begin{aligned} \max \quad & \textstyle\sum_{i \in V} x_i \\ \text{s.t. } & x_i + x_j \leq 1 \ (\{i,j\} \in E) && (1) \\ & x \in \{0,1\}^n, && (2) \end{aligned}$$

where the variable x_i takes the value 1 if and only if vertex i is in the stable set.

The convex hull in $\mathbb{R}^n$ of feasible solutions to (1)–(2) is called the *stable set polytope* and denoted by STAB(G). This polytope has been studied in great depth [4, 11, 21]. The most well-known facet-defining inequalities for STAB(G) are the *clique* inequalities of Padberg [21]. A *clique* in G is a set of pairwise adjacent vertices, and the associated inequalities take the form:

$$\sum_{i \in C} x_i \leq 1 \qquad (\forall C \in \mathcal{C}), \tag{3}$$

where $\mathcal{C}$ denotes the set of maximal cliques in G. Note that the clique inequalities dominate the edge inequalities (1).

The separation problem for clique inequalities is $\mathcal{NP}$-hard [20]. Fortunately, some fast and effective separation heuristics exist, not only for clique inequalities, but also for various other inequalities (e.g., [4, 23, 25]). Nevertheless, LP-based approaches can run into difficulties when n exceeds 200, mainly due to the weakness of the upper bounds.

2.2 Semidefinite Programming Relaxations

Lovász [17] introduced an upper bound for the SSP, called the *theta number* and denoted by $\theta(G)$, which is based on an SDP relaxation. The bound can be derived

in several different ways, and we follow the derivation presented in [11]. We start by formulating the SSP as the following non-convex quadratically-constrained program:

$$\begin{aligned} \max \quad & \textstyle\sum_{i \in V} x_i && (4)\\ \text{s.t. } & x_i^2 - x_i = 0 \ (i \in V) && (5)\\ & x_i x_j = 0 \quad (\{i,j\} \in E). && (6) \end{aligned}$$

In order to linearise the constraints, we introduce an auxiliary matrix variable $X = xx^T$, along with the augmented matrix

$$Y = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^T = \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix}.$$

We then note that Y is real, symmetric and positive semidefinite (psd), which we write as $Y \succeq 0$. This leads to the following SDP relaxation:

$$\begin{aligned} \max \quad & \textstyle\sum_{i \in V} x_i && (7)\\ \text{s.t. } & x = \operatorname{diag}(X) && (8)\\ & X_{ij} = 0 \quad (\{i,j\} \in E) && (9)\\ & Y \succeq 0. && (10) \end{aligned}$$

Solving this SDP yields $\theta(G)$.

In practice, $\theta(G)$ is often a reasonably strong upper bound for the SSP (e.g., [3, 6, 12, 23]). Unfortunately, solving large-scale SDPs can be rather time-consuming, which makes SDP relaxations somewhat unattractive for use within a branch-and-bound framework.

The above SDP can be strengthened by adding various valid inequalities (e.g., [4, 6, 10, 12, 18, 26]). We omit details, for the sake of brevity.

2.3 The Lovász Theta Body and Ellipsoids

The following beautiful result can be found in Grötschel, Lovász & Schrijver [11]. Let us define the following polytope:

$$\mathrm{QSTAB}(G) = \left\{ x \in \mathbb{R}^n_+ : (3) \text{ hold} \right\},$$

along with the following convex set:

$$\mathrm{TH}(G) = \left\{ x \in \mathbb{R}^n : \exists X \in \mathbb{R}^{n \times n} : (8) - (10) \text{ hold} \right\}.$$

Then we have:

$$\mathrm{STAB}(G) \subseteq \mathrm{TH}(G) \subseteq \mathrm{QSTAB}(G).$$

This implies that $\theta(G)$ dominates the upper bound for the SSP based on clique inequalities.

The set $\mathrm{TH}(G)$ is called the *theta body*. In [11], a characterisation of $\mathrm{TH}(G)$ is given in terms of linear inequalities. We are interested here, however, in a

characterisation of TH(G) in terms of convex quadratic inequalities, due to Fujie & Tamura [9]. For a given vector $\mu \in \mathbb{R}^m$, let $M(\mu)$ denote the symmetric matrix with $\mu_{ij}/2$ in the ith row and jth column whenever $\{i,j\} \in E$, and zeroes elsewhere. Then, given vectors $\lambda \in \mathbb{R}^n$ and $\mu \in \mathbb{R}^m$ such that $\mathrm{Diag}(\lambda) + M(\mu)$ is psd, the set

$$E(\lambda,\mu) = \left\{x \in \mathbb{R}^n : \; x^T(\mathrm{Diag}(\lambda) + M(\mu))x \leq \lambda^T x\right\}$$

is easily shown to be an ellipsoid that contains STAB(G). The result we need is the following:

Theorem 1 (Fujie & Tamura, 2002). *For any graph G, we have:*

$$TH(G) = \bigcap_{\lambda,\mu:\mathrm{Diag}(\lambda)+M(\mu)\succeq 0} E(\lambda,\mu).$$

2.4 Relaxations of Non-convex Quadratic Problems

For what follows, we will also need a known result concerning relaxations of non-convex quadratic problems, mentioned for example in [8, 16, 22]. Given a problem of the form:

$$\begin{aligned} \inf \quad & x^T Q^0 x + c^0 \cdot x \\ \text{s.t. } & x^T Q^j x + c^j \cdot x = b_j \; (j = 1,\ldots,r) \\ & x \in \mathbb{R}^n, \end{aligned} \tag{11}$$

we can construct the following SDP relaxation:

$$\begin{aligned} \inf \quad & Q^0 \bullet X + c^0 \cdot x \\ \text{s.t. } & Q^j \bullet X + c^j \cdot x = b_j \; (j = 1,\ldots,r) \\ & Y \succeq 0, \end{aligned} \tag{12}$$

where $Q^j \bullet X$ denotes $\sum_{i=1}^n \sum_{k=1}^n Q^j_{ik} X_{ik}$. Alternatively, we can form a Lagrangian relaxation by relaxing the constraints (11), using a vector $\phi \in \mathbb{R}^r$ of Lagrangian multipliers. The relaxed problem is then to minimise

$$f(x,\lambda) \; = \; x^T \left(Q^0 + \sum_{j=1}^r \phi_j Q^j\right) x + \left(c^0 + \sum_{j=1}^r \phi_j c^j\right) \cdot x - \sum_{j=1}^r \phi_j b_j$$

subject to $x \in \mathbb{R}^n$.

The result that we need is as follows:

Theorem 2 (Various authors). *Suppose the SDP satisfies the Slater condition, so that an optimal dual solution to the SDP exists. Then optimal Lagrangian multipliers ϕ^* exist too, and are nothing but the optimal dual vectors for the constraints (12) in the SDP. Moreover, the function $f(x,\phi^*)$ is convex, and therefore the matrix $Q^0 + \sum_{j=1}^r \phi_j Q^j$ is psd.*

An analogous result holds when quadratic inequalities, rather than equations, are present: one simply makes the associated multipliers non-negative.

3 The New Approach

In this section, the new approach is described.

3.1 An 'Optimal' Ellipsoid

Recall that Theorem 1 expresses $\mathrm{TH}(G)$ as the intersection of an infinite family of ellipsoids. The following proposition states that one can efficiently compute a particular ellipsoid with a very desirable property.

Theorem 3. *Let λ^* and μ^* be optimal dual vectors for the constraints (8) and (9) in the SDP (7)-(10). Then:*

$$\theta(G) = \max\left\{\sum_{i\in V} x_i : x \in E(-\lambda^*, -\mu^*)\right\}.$$

Proof. The SDP (7)-(10) has the form specified in Theorem 3.1 of Tunçel [27], and therefore satisfies the Slater condition. As a result, the optimal dual solution (λ^*, μ^*) exists and its cost is equal to $\theta(G)$. Applying Theorem 2, but switching signs to take into account the fact that the SDP is of maximisation type, we have:

$$\theta(G) = \max\left\{\sum_{i\in V} x_i - x^T(\mathrm{Diag}(\lambda^*) + M(\mu^*))x + \lambda^* \cdot x :\ x \in \mathbb{R}^n\right\}.$$

Moreover, the matrix $-\mathrm{Diag}(\lambda^*) - M(\mu^*)$ must be psd.

Now, given that this is a concave maximisation problem, and the fact that the pair (λ^*, μ^*) form an optimal set of Lagrangian multipliers, we have:

$$\theta(G) = \max\left\{\sum_{i\in V} x_i :\ -x^T(\mathrm{Diag}(\lambda^*) + M(\mu^*))x \le -\lambda^* \cdot x,\ x \in \mathbb{R}^n\right\},$$

which proves the result. □

In other words, the dual solution to the SDP can be used to construct a relaxation of the SSP that has a linear objective function and a single convex quadratic constraint, whose corresponding upper bound is equal to $\theta(G)$.

Example: Let G be the 5-hole, i.e., a chordless cycle on 5 nodes. The optimal dual solution has $\lambda_i^* = -1$ for all i and $\mu_e^* = (1-\sqrt{5})/2$ for all e. The corresponding ellipsoid is:

$$\left\{x \in \mathbb{R}^n : \sum_{i\in V} x_i^2 + (\sqrt{5}-1)\sum_{\{i,j\}\in E} x_i x_j \le \sum_{i\in V} x_i\right\}.$$

Although it is hard to visualise this ellipsoid, one can show that all points in it satisfy the linear inequality:

$$\sum_{i\in V} x_i \le \sqrt{5}.$$

This agrees with the known fact that, for the 5-hole, $\theta(G) = \sqrt{5}$ [11]. □

3.2 Cutting Planes from the Optimal Ellipsoid

One way in which to exploit the existence of the optimal ellipsoid would be to construct a branch-and-bound algorithm in which convex quadratically-constrained programs are solved at each node of the enumeration tree. For example, one could solve at each node a relaxation of the form:

$$\begin{aligned} \max \quad & \textstyle\sum_{i \in V} x_i \\ \text{s.t. } & x \in E(-\lambda^*, -\mu^*) \\ & \textstyle\sum_{i \in C} x_i \leq 1 \quad (C \in \mathcal{C}') \\ & x \in [0,1]^n, \end{aligned}$$

where $\mathcal{C}'$ is a suitably chosen collection of clique inequalities.

Here, however, we are concerned with the use of the optimal ellipsoid to generate cutting planes (violated valid linear inequalities), to be used within an LP-based cut-and-branch or branch-and-cut algorithm. In this subsection, we consider the (relatively) easy case in which the cutting planes are simply tangent hyperplanes to the optimal ellipsoid. A way to strengthen these cutting planes will be presented in the following subsection.

As before, let the optimal dual solution be (λ^*, μ^*). Our experience is that the matrix $\mathrm{Diag}(\lambda^*) + M(\mu^*)$ is invariably negative definite in practice, which indicates that the ellipsoid $E(-\lambda^*, -\mu^*)$ is bounded. Under this condition, the matrix is invertible and the optimal ellipsoid is easily shown to have the unique central point $\hat{x} = \frac{1}{2}(\mathrm{Diag}(\lambda^*) + M(\mu^*))^{-1}\lambda^*$.

Observe that the dual solution (λ^*, μ^*) and the centre $\hat{x}$ can be computed once and for all, as a kind of 'pre-processing' step, and stored in memory. Then, to generate cutting planes, one can use the following separation algorithm:

1. Let $x^* \in [0,1]^n$ be the current fractional point to be separated, and suppose that $x^* \notin E(-\lambda^*, -\mu^*)$.
2. Perform a line-search to find a point $\tilde{x}$, that is a convex combination of x^* and $\hat{x}$ and lies on the boundary of $E(-\lambda^*, -\mu^*)$.
3. Using a first-order Taylor approximation, find a linear inequality $a^T x \leq b$ that is both violated by x^*, and defines a tangent hyperplane to $E(-\lambda^*, -\mu^*)$ at $\tilde{x}$.
4. Scale the vector a and scalar b by a constant factor Δ and round down to the nearest integers.

More details will be given in the full version of this paper. We remark that the last step helps to avoid numerical difficulties due to rounding errors, and also makes it easier to strengthen the inequality, as described in the next subsection.

3.3 Cut Strengthening

Let $\alpha^T x \leq \beta$, with $\alpha \in \mathbb{Z}^n$ and $\beta \in \mathbb{Z}_+$, be a cutting plane generated by the procedure described in the previous subsection. To strengthen it, we examine one

variable at a time, and solve a pair of small optimisation problems. Specifically, given a variable x_j, we compute for $k \in \{0, 1\}$ the value:

$$\gamma_j^k = \max \left\{ \sum_{i \neq j} \alpha_i x_i : \; x \in E(-\lambda^*, -\mu^*), \; x_j = k \right\}, \tag{13}$$

and then the value

$$\delta_j^k = \min \left\{ \beta, \lfloor \gamma_j^k \rfloor \right\}.$$

By construction, δ_j^k is an upper bound on the value taken by $\sum_{i \neq j} \alpha_i x_i$ in any feasible solution, subject to the constraint $x_j = k$. As a result, we can replace the original right-hand side β with δ_j^0 and change the coefficient of x_j from α_j to $\delta_j^0 - \delta_j^1$.

This strengthening procedure, which can be regarded as a special case of a procedure described in [2], can be applied to any set of variables, in any desired sequence. Different choices for the set and the sequence may lead to different cutting planes, starting from the same tangent hyperplane $\alpha^T x \leq \beta$. Again, details will be given in the full version of the paper.

The overall procedure described in this subsection and the last will be referred to as the *Ellipsoid Cut Generation Algorithm* (ECGA).

4 Computational Experiments

In order to gain some insight into the potential effectiveness of the 'ellipsoidal' approach, we have designed and coded a rudimentary LP-based 'cut-and-branch' framework, in which cutting planes are used to strengthen an initial 0-1 LP formulation, and the strengthened formulation is then fed into a standard branch-and-bound solver (see, e.g., [20]).

4.1 The Cut-and-Branch Algorithm

The initial 0-1 LP formulation contains a family of clique inequalities associated with a *clique-cover* of G, by which we mean a set of cliques such that each edge $\{i, j\} \in E$ is contained in at least one clique in the family (see, e.g., [3]). This clique-cover is computed by a simple greedy algorithm. The LP relaxation is then solved, and a cutting-plane algorithm based clique inequalities is then run. The separation problem for clique inequalities is solved by a greedy heuristic. We denote by $\mathrm{UB}_{\mathrm{clique}}$ the upper bound obtained when the clique separator fails.

At that point, a cutting-plane algorithm based on ECGA is executed. In the current implementation of ECGA, the parameter Δ is set to 10^4, and several strengthened cutting planes are generated from each original cutting plane. This is done by strengthening in turn the first k variables, then the second k, and so on (k is always chosen in the range $[3, 10]\%$ of n, except in the case of the `p_hat` graphs, as explained below). This choice is motivated by the empirical

observation that it is usually the first few coefficient strengthenings that are the most important. We denote by $\mathrm{UB}_{\mathrm{ellips}}$ the upper bound obtained at the root node at completion of the ECGA-based cutting-plane algorithm.

The strengthening subproblems (13) are solved by the CPLEX `baropt` algorithm. (As pointed out by a referee, they could perhaps be solved directly with linear algebra, which would probably be much faster.) The strengthening step turned out to be of particular relevance, as the 'raw' ellipsoidal cuts described in Subsection 3.2 often turned out to be dense, have 'nasty' coefficients, and be nearly parallel to the objective function. Therefore, cut strengthening leads to better numerical stability.

The computations were run on a workstation equipped with 2 Intel Xeon 5150 processors clocked at 2.66 GHz and with 4GB of RAM, under the Linux 64bit operating system. However, all experiments were performed in single thread mode. The algorithm was implemented within the IBM CPLEX 11.2 framework, in which cuts are added by the `usercuts` option. All CPLEX parameters were left at their default settings, apart from the `branching direction`, which was set to `up`, and the `mip emphasis`, which was set to `optimality`.

4.2 Preliminary Computational Results

In this subsection, we give empirical evidence of the effectiveness of the cuts generated from the ellipsoid. The experiments are based on the instances from the DIMACS Second Challenge (Johnson & Trick [13]) with $n < 400$, available at the web site [5]. Among all such graphs, we consider only those instances for which $\lfloor \mathrm{UB}_{\mathrm{clique}} \rfloor > \alpha(G)$. The corresponding optimal ellipsoids $E(-\lambda^*, -\mu^*)$ have been computed by the SDP solver [19] (coded in Matlab) available at [1].

In Table 1 we report for each graph the name, the number of nodes n and edges m, the cardinality of the maximum stable set $\alpha(G)$, the Lovász theta number $\theta(G)$, and the upper bounds $\mathrm{UB}_{\mathrm{clique}}$ and $\mathrm{UB}_{\mathrm{ellips}}$ mentioned in the previous subsection.

In Table 2, a comparison between two cut-and-branch algorithms, one based only on clique inequalities and the other embedding also cuts from the ellipsoid, is presented. It is important to remark that the first cut-and-branch algorithm is in itself rather competitive. Indeed, it often outperforms the dedicated branch-and-cut algorithms described in [23] and and [25], even though they both benefit from having several separation routines, dedicated cut pool management, and specialised branching strategies.

For each algorithm, the number of evaluated subproblems, as well as the total CPU time (in seconds), are reported. In the case of the ellipsoid cut-and-branch two more columns show how the total CPU time is split between branch-and-bound phase and cut lifting, while the total separation time from $E^*(-\lambda^*, -\mu^*)$ (Step 1-3 of ECGA) is always negligible (less than 0.1 secs for largest instances) and is not explicitly reported. The last column in the table contains the number of cuts generated by ECGA.

Table 1 shows that our approach always returns upper bounds very close to $\theta(G)$. To our knowledge, this has not been achieved before using cutting planes

Table 1. Root upper bounds

Graph name	n	m	$\alpha(G)$	$\theta(G)$	$\text{UB}_{\text{clique}}$	$\text{UB}_{\text{ellips}}$
brock200_1	200	5,066	21	27.50	38.20	27.79
brock200_2	200	10,024	12	14.22	21.53	14.32
brock200_3	200	7,852	15	18.82	27.73	19.00
brock200_4	200	6,811	17	21.29	30.84	21.52
C.125.9	125	787	34	37.89	43.06	38.05
C.250.9	250	3,141	44	56.24	72.04	57.41
c-fat200-5	200	11,427	58	60.34	66.66	60.36
DSJC125.1	125	736	34	38.39	43.15	38.44
DSJC125.5	125	3,891	10	11.47	15.60	11.48
mann_a27	378	702	126	132.76	135.00	132.88
keller4	171	5,100	11	14.01	14.82	14.09
p_hat300-1	300	33,917	8	10.10	17.71	10.15
p_hat300-2	300	22,922	25	27.00	34.01	27.14
p_hat300-3	300	11,460	36	41.16	54.36	41.66
san200_0.7-2	200	5,970	18	18.00	20.14	18.10
sanr200_07	200	6,032	18	23.80	33.48	24.00
sanr200_09	200	2,037	42	49.30	60.04	49.77

Table 2. Cut-and-branch results

	Clique cut-and-branch		Ellipsoid cut-and-branch				
Graph name	#sub.	B&b time	#sub.	Lifting time	B&b time	Total time	#cuts
brock200_1	270,169	1,784.78	122,387	75.48	1,432.76	1,508.24	40
brock200_2	6,102	83.16	2,689	109.27	209.39	318.66	30
brock200_3	52,173	459.02	7,282	117.38	252.17	369.55	30
brock200_4	85,134	735.52	18,798	180.83	468.52	649.35	40
C.125.9	3,049	5.63	2,514	17.37	6.52	23.89	75
C.250.9	—	—	—	634.23	—	—	250
c-fat200-5	47	12.06	47	51.35	25.16	76.51	10
DSJC125.1	4,743	6.13	2,981	17.92	8.09	26.01	75
DSJC125.5	1,138	6.97	369	18.29	9.66	27.95	50
mann a27	4,552	2.06	1,278	15.60	2.46	18.06	10
keller4	4,856	21.88	3,274	45.21	36.55	81.76	34
p hat300-1	4,518	124.36	4,238	14.56	132.54	147.10	6
p hat300-2	7,150	194.27	1,522	66.59	204.88	271.47	10
p hat300-3	398,516	10,270.53	107,240	99.43	8,756.76	8,856.19	15
san200 0.7-2	86	8.58	0	66.23	0.76	66.99	10
sanr200_07	88,931	827.52	42,080	116.34	733.47	849.81	60
sanr200_09	635,496	2,650.55	274,261	158.64	1,281.52	1,440.16	100

— time limit reached

that involve only the x variables. (It may be asked why the bound $\text{UB}_{\text{ellips}}$ is not *superior* to $\theta(G)$, in light of Theorem 3. This is due to numerical issues, tailing off, and our chosen time limit.) The great potential of the cuts is

confirmed by the cut-and-branch results of Table 2. Note that, for all instances, the number of evaluated subproblems decreases significantly (with the exception of `c-fat-200-5`) when cuts from the ellipsoid are generated. Moreover, this is accomplished by a fairly small number of cuts. Thus, although the cuts tend to be dense, the simplex algorithm does not appear to be particularly slowed down.

Notice also that in 5 instances out of 17, namely the largest and most difficult ones, ECGA is cost-effective, as the time spent in cut generation is more than compensated by the reduction in the size of the enumeration tree. It is worthwhile mentioning that, for the `p_hat` graphs, only one coefficient needed to be strengthened in order to obtain good cuts.

5 Concluding Remarks

The key idea in this paper is that one can use the dual solution to the SDP relaxation to construct an ellipsoid that wraps reasonably tightly around the stable set polytope, and that this ellipsoid can be used to construct quite strong cutting planes in the original (linear) space. The computational results, though preliminary, indicate that this approach is promising.

There is, however, room for further improvement. In particular, there are several ways in which the ECGA could be modified or extended. For example:

- Instead of using the 'optimal' ellipsoid to generate cutting planes, one could use some other ellipsoid, or indeed a whole family of ellipsoids. This raises the question of how to generate one or more ellipsoids in such a way that one obtains deep cutting planes, but without excessive computing times.
- In the cut-strengthening procedure, one might be able to obtain smaller coefficients γ_j^k by including some valid linear inequalities (such as clique inequalities) in the maximisation (13). This would lead to stronger cutting planes, but the time taken to compute the coefficients would increase.
- One could also search for effective heuristic rules for ordering the variables in the cut-strengthening step.
- As mentioned in Subsection 2.2, the SDP relaxation (7)-(10) can be strengthened by adding valid inequalities. (For example, Schrijver [26] suggested adding the inequalities $X_{ij} \geq 0$ for all $\{i, j\} \notin E$.) Let $\tilde{\theta}(G) < \theta(G)$ be an improved upper bound obtained by solving such a strengthened SDP. The proof of Theorem 3 can be modified to show the existence of an ellipsoid, say $\tilde{E}$, such that

$$\tilde{\theta}(G) = \max \left\{ \sum_{i \in V} x_i : x \in \tilde{E} \right\}.$$

  Such an ellipsoid might produce stronger cutting planes.

Progress on these issues, if any, will be reported in the full version of the paper.

Finally, one could explore the possibility of adapting the ellipsoidal approach to other combinatorial optimisation problems. In our view, this is likely to work

well only for problems that have a 'natural' formulation as a continuous optimisation problem with a linear objective function and non-convex quadratic constraints, like the formulation (4)–(6) of the SSP.

Acknowledgments. The second author was supported by the Engineering and Physical Sciences Research Council under grant EP/D072662/1. Thanks are due to an anonymous referee whose comments helped to improve the paper.

References

1. Alpen-Adria-Universität Klagenfurt website, http://www.math.uni-klu.ac.at/or/Software
2. Andersen, K., Pochet, Y.: Coefficient strengthening: a tool for reformulating mixed-integer programs. Math. Program. 122, 121–154 (2009)
3. Balas, E., Ceria, S., Cornuéjols, G., Pataki, G.: Polyhedral methods for the maximum clique problem. In: Johnson, D.S., Trick, M.A. (eds.) Cliques, Coloring and Satisfiability, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, pp. 11–28 (1996)
4. R.: Borndörfer, Aspects of Set Packing, Partitioning and Covering. Doctoral Thesis, Technical University of Berlin (1998)
5. DIMACS Repository, ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique
6. Dukanovic, I., Rendl, F.: Semidefinite programming relaxations for graph coloring and maximal clique problems. Math. Program 109, 345–365 (2007)
7. Fahle, T.: Simple and Fast: Improving a Branch-And-Bound Algorithm for Maximum Clique. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 47–86. Springer, Heidelberg (2002)
8. Fujie, T., Kojima, M.: Semidefinite programming relaxation for nonconvex quadratic programs. J. Glob. Opt. 10, 367–380 (1997)
9. Fujie, T., Tamura, A.: On Grötschel-Lovász-Schrijver's relaxation of stable set polytopes. J. Oper. Res. Soc. Japan 45, 285–292 (2002)
10. Giandomenico, M., Letchford, A.N.: Exploring the relationship between max-cut and stable set relaxations. Math. Program 106, 159–175 (2006)
11. Grötschel, M., Lovász, L., Schrijver, A.J.: Geometric Algorithms in Combinatorial Optimization. Wiley, New York (1988)
12. Gruber, G., Rendl, F.: Computational experience with stable set relaxations. SIAM J. Opt. 13, 1014–1028 (2003)
13. Johnson, D.S., Trick, M.A. (eds.): Cliques, Coloring and Satisfiability: the 2nd DIMACS Implementation Challenge. American Mathematical Society, Providence
14. Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math. 182, 105–142 (1999)
15. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum, New York (1972)
16. Lemaréchal, C., Oustry, F.: SDP relaxations in combinatorial optimization from a Lagrangian viewpoint. In: Hadjisawas, N., Pardalos, P.M. (eds.) Advances in Convex Analysis and Global Optimization. Kluwer, Dortrecht (2001)
17. Lovász, L.: On the Shannon capacity of a graph. IEEE Trans. Inform. Th. IT-25, 1–7 (1979)

18. Lovász, L., Schrijver, A.J.: Cones of matrices and set-functions and 0-1 optimization. SIAM J. Optimization 1, 166–190 (1991)
19. Malick, J., Povh, J., Rendl, F., Wiegele, A. (2007) Boundary Point Method for solving SDPs: mprw.m, Inst. f. Mathematik, Alpen-Adria-Universität Klagenfurt (2007)
20. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
21. Padberg, M.W.: On the facial structure of set packing polyhedra. Math. Program. 5, 199–215 (1973)
22. Poljak, S., Rendl, F., Wolkowicz, H.: A recipe for semidefinite relaxation for (0,1)-quadratic programming. J. Global Opt. 7, 51–73 (1995)
23. Rebennack, S., Oswald, M., Theis, D.O., Seitz, H., Reinelt, G., Pardalos, P.M.: A branch and cut solver for the maximum stable set problem. J. Comb. Opt. (2010) (to appear)
24. Régin, J.-C.: Solving the maximum clique problem with constraint programming. In: Proceedings of CPAIOR 2003. LNCS, vol. 2883, pp. 634–648. Springer, Heidelberg (2003)
25. Rossi, F., Smriglio, S.: A branch-and-cut algorithm for the maximum cardinality stable set problem. Oper. Res. Lett. 28, 63–74 (2001)
26. Schrijver, A.J.: A comparison of the Delsarte and Lovász bounds. IEEE Trans. Inf. Th. IT-25, 425–429 (1979)
27. Tunçel, L.: On the Slater condition for SDP relaxations of nonconvex sets. Oper. Res. Lett. 29, 181–186 (2001)

Capacitated Vehicle Routing with Non-uniform Speeds

Inge Li Gørtz[1], Marco Molinaro[2,⋆], Viswanath Nagarajan[3], and R. Ravi[4,*]

[1] Technical University of Denmark
[2] Tepper School of Business, Carnegie Mellon University
[3] IBM T.J. Watson Research Center
[4] Tepper School of Business, Carnegie Mellon University

Abstract. The *capacitated vehicle routing problem* (CVRP) [21] involves distributing (identical) items from a depot to a set of demand locations in the shortest possible time, using a single capacitated vehicle. We study a generalization of this problem to the setting of multiple vehicles having non-uniform speeds (that we call *Heterogenous CVRP*), and present a constant-factor approximation algorithm.

The technical heart of our result lies in achieving a constant approximation to the following TSP variant (called *Heterogenous TSP*). Given a metric denoting distances between vertices, a depot r containing k vehicles having speeds $\{\lambda_i\}_{i=1}^{k}$, the goal is to find a tour for each vehicle (starting and ending at r), so that every vertex is covered in some tour and the maximum completion time is minimized. This problem is precisely Heterogenous CVRP when vehicles are uncapacitated.

The presence of non-uniform speeds introduces difficulties for employing standard tour-splitting techniques. In order to get a better understanding of this technique in our context, we appeal to ideas from the 2-approximation for minimum makespan scheduling in unrelated parallel machines of Lenstra et al. [19]. This motivates the introduction of a new approximate MST construction called *Level-Prim*, which is related to *Light Approximate Shortest-path Trees* [18]. The last component of our algorithm involves partitioning the Level-Prim tree and matching the resulting parts to vehicles. This decomposition is more subtle than usual since now we need to enforce correlation between the lengths of the parts and their distances to the depot.

1 Introduction

The capacitated vehicle routing problem (CVRP) is an extensively studied combinatorial optimization problem (see e.g., [21] and references therein). CVRP is defined on a metric space (V, d), where V is a finite set of locations/vertices and $d : V \times V \to \mathbb{R}_+$ a distance function that is symmetric and satisfies triangle inequality. There is a depot vertex $r \in V$ that contains an infinite supply of an identical item, and each vertex $u \in V$ demands some units q_u of this item. A

⋆ Supported in part by NSF grant CCF-0728841.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 235–247, 2011.

single vehicle of integral capacity $Q > 0$ is used to distribute the items. The objective is to find a minimum length tour of the vehicle that satisfies all demands subject to the constraint that the vehicle carries at most Q units at any time.

CVRP is closely related to the Traveling Salesman Problem (TSP). It is clear that CVRP reduces to TSP in the absence of capacity constraint. More interestingly, a reverse relation is also known—essentially the best known approximation algorithm for CVRP [17] achieves a guarantee of $\rho + 1$, where ρ is the best approximation ratio for TSP.

In practice, it is natural to have a fleet of *multiple* vehicles that can run in parallel. The objective can then be to either minimize the sum of completion times of all the vehicles or to minimize the maximum completion time over all vehicles (or the makespan of the routing). Furthermore the vehicles can all be identical (same speed) or heterogeneous (have different speeds). In either case, it is not hard to see that the total completion time objective reduces to the usual CVRP on a single maximum-speed vehicle, and constant-factor approximation algorithms readily follow.

When the objective is to minimize the makespan with identical vehicles, ideas for approximating the regular CVRP problem using a tour-splitting heuristic introduced by Frederickson et al. [14] can be easily adapted to derive a constant-factor approximation algorithm (see below).

This motivates HetCVRP, the *Heterogenous Capacitated Vehicle Routing Problem* that we consider. Here, a fleet of k vehicles with *non-uniform speeds* and uniform capacities is initially located at the depot vertex r. The objective is to satisfy the demands subject to the capacity constraints while minimizing the makespan. Our main result is a constant-factor approximation algorithm for HetCVRP.

Most of our algorithmic ideas lie in solving the special case of HetCVRP when there is no capacity constraint. This problem, which we call HetTSP, is a generalization of TSP that might be of independent interest. For most of this paper, we will focus on obtaining a constant-factor approximation for HetTSP.

1.1 Previous Techniques

Tour-splitting solutions: To illustrate the use of known techniques, we outline how to obtain a constant-factor approximation algorithm for HetTSP with uniform speeds [14]. First, notice that the union of the tours of OPT connects all vertices, and hence a minimum spanning tree (MST) has length at most $k \cdot \mathsf{OPT}$. Then consider a MST, duplicate its edges and take an Euler tour C, which is of length $d(C) \leq 2k \cdot \mathsf{OPT}$. Now split C into k segments of lengths at most $\frac{d(C)}{k}$ by removing edges. Finally, the tour for the i^{th} vehicle is obtained by connecting both endpoints of the i^{th} segment of C to the depot. Since twice the distance from the depot to any vertex is a lower bound on OPT, the length of each tour is at most $3 \cdot \mathsf{OPT}$ and hence this solution is a 3-approximation. We remark that this can be extended to obtain an $O(1)$-approximation for HetCVRP with uniform speeds (e.g., using Theorem 2).

At a very high level, this strategy has two main components: (1) Partitioning an MST into manageable-sized connected parts; (2) assigning these parts to vehicles. This simple idea—which was already present in the 70's—is the central piece of many heuristics and approximations for vehicle routing problems (e.g., [14,17,13,3,16,1,2]). However, it is not clear how to employ this technique in the presence of vehicles with multiple speeds. This is because the two main components now need some correlation: a small part of the MST, which should be assigned to a slower vehicle, must also be relatively closer to the depot in order to be reachable by this vehicle.

Set-cover based solutions: For HetTSP with non-uniform speeds, previous approaches seem to give only a logarithmic approximation, as follows. Guess the optimal makespan OPT (within a constant factor). If each vehicle of speed s is given a length budget of $s \cdot \mathsf{OPT}$, then the vehicles can collectively cover all vertices. Using an approximation algorithm for k-MST [15] (or the related orienteering problem [6,8]) within a maximum-coverage framework (see e.g.. [9]), we can obtain tours of length OPT that cover a constant fraction of all vertices. Repeating this coverage step until all vertices are covered gives a solution to HetTSP of makespan $O(\log n) \cdot \mathsf{OPT}$. The intrinsic problem of this approach is that it is too general—in fact, the above algorithm also yields a logarithmic approximation even in the setting where the metric faced by each vehicle is arbitrary (instead of just scaling by its speed), and this generalization of HetTSP can be shown to be set-cover hard. It is unclear whether the approximation of this set-covering based approach can be improved for HetTSP.

1.2 Results, Techniques and Outline

We extend the tour-splitting approach described above to obtain the following result.

Theorem 1. *There are constant-factor approximation algorithms for* HetTSP *and* HetCVRP.

In order to obtain the approximation for HetTSP, we abstract the requirements of the two components in the tour-splitting strategy. As a preprocessing step, we round the speeds of vehicles to powers of two and guess the optimum makespan M. First, we specify conditions which guarantee that a collection of r-rooted trees is "assignable", that is, each vehicle can visit the nodes of the trees assigned to it within time $O(M)$ (Definition 1). The conditions in Definition 1 are based on the LP to obtain a 2-approximation for *scheduling in unrelated parallel machines* by Lenstra et al. [19].

Secondly, instead of partitioning an MST as in the previous section, we consider more structured spanning trees which we call Level-Prim trees. Consider grouping the vertices ordered according to their distance from r into levels, where the ith level includes all vertices within distance $2^i M$.[1] The Level-Prim

[1] Notice that given the rounding of vehicle speeds to powers of two, vertices in level i can only be served by vehicles of speed 2^i or higher given the makespan bound M.

tree is simply the tree resulting from running Prim's algorithm with the restriction that all nodes in a level are spanned before starting to pull in nodes from the next. A Level-Prim tree has two important properties: (i) The vertices along every root-leaf path are monotonically nondecreasing in level and (ii) For every suffix of levels, the subgraph induced on it costs at most $O(1)$ times its induced MST. The first condition, which is the departing point from MSTs, greatly simplifies the decomposition procedure carried in the next step. The second property guarantees that we can decompose a Level-Prim tree into an assignable collection. These properties are formalized in Theorem 3.

The Level-Prim construction combines both MST and shortest-path distances from a root, so it is not surprising that this structure is related to *Light Approximate Shortest-Path Trees* (LAST) introduced by Khuller et al. [18]. Indeed, we use the existence of a suitably defined LAST in proving Theorem 3. We remark, however, that the properties guaranteed by LASTs are not enough for our purposes (see Sect. 3.2).

The third main component of our approximation for HetTSP is decomposing Level-Prim into an assignable collection of r-rooted trees. Roughly, we partition the edges of Level-Prim into subtrees while ensuring that each subtree consisting of vertices in levels up to i (and hence is at a distance of about $2^i M$ from the root) also has total length approximately $2^i M$, and thus can be assigned to a vehicle of speed about 2^i. This partition, which relies on the two properties of Level-Prim, gives a collection of *unrooted* trees which is assignable. Due to the length of these trees, the extra distance to connect them to the root r can be charged to their edges, hence this collection can be turned into a r-rooted assignable collection.

Heterogeneous CVRP: Recall that the input to the *Heterogenous CVRP* (HetCVRP) is the same as that for HetTSP with an additional vehicle capacity Q. We study the "split-delivery" version of CVRP here, where demand at a vertex may be served by multiple visits to it; however, our result easily extends to the "unsplit-delivery" HetCVRP. We show that the HetCVRP problem can be reduced to HetTSP in an approximation preserving way; so we also obtain an $O(1)$-approximation for HetCVRP.

The idea in this reduction is to modify the input metric based on the connectivity and capacitated routing lower-bounds for CVRP [17]. The modified distance function encodes any additional trips to and from the root that a vehicle has to make if it runs out of capacity. Let $G = (V, E)$ denote the complete graph on vertices V with edge-weights equal to distances d. Augment G to a new graph H by adding vertices $V' = \{v_p : v \in V, p \in [q_v]\}$, and edges $E' = \{(v, v_p) : v \in V, p \in [q_v]\}$; each edge (v, v_p) has weight $\frac{d(r,v)}{Q}$. For any vertex $v \in V$, the vertices $\{v_p : p \in [q_v]\}$ are referred to as copies of v. Let (V', ℓ) denote the metric induced on vertices V' where ℓ denotes the shortest-path distances in graph H. We let $\mathcal{J}$ be the instance of HetTSP on metric (V', ℓ) with depot r and k vehicles having speeds $\{\lambda_i\}_{i=1}^k$. We can prove the following based on the above reduction (details omitted).

Theorem 2. *Consider an instance $\mathcal{I}$ of* HetCVRP. *There is a poly-time constructible instance $\mathcal{J}$ of* HetTSP *such that* $\mathsf{OPT}_{\mathsf{tsp}}(\mathcal{J}) = O(1) \cdot \mathsf{OPT}_{\mathsf{vrp}}(\mathcal{I})$. *Moreover, a solution to $\mathcal{J}$ of makespan M can be converted in poly-time to a solution to $\mathcal{I}$ with makespan $O(M)$.*

1.3 Related Work

For the CVRP, the best known approximation ratio [17] is essentially $\rho+1$ where ρ is the best guarantee for TSP. The current best values for ρ are $\rho = \frac{3}{2}$ for general metrics [10], and $\rho = 1 + \epsilon$ (for any constant $\epsilon > 0$) for constant dimensional Euclidean metrics [4,20]. This has been improved slightly to $1 + \rho \cdot (1 - \frac{1}{Q}) - \frac{1}{3Q^3}$ when $Q \geq 3$ [7]. Recently, Das and Mathieu [12] gave a quasi-polynomial time approximation scheme for CVRP on the Euclidean plane.

Several variants of TSP have been studied, most of which have a min-sum objective. One related problem with min-max objective is *nurse station location* [13], where the goal is to obtain a collection of trees (each rooted at a distinct depot) such that all vertices are covered and the maximum tree length is minimized. Even et al. [13] gave a 4-approximation algorithm for this problem, based on partitioning the MST and assigning to depots along the lines of Sect. 1.1; their second step, however, involves a non-trivial bipartite matching subproblem.

In proving the properties of Level-Prim, we use *Light Approximate Shortest-Path Trees* introduced by Khuller, Raghavachari and Young [18], building on the work on shallow-light trees of Awerbuch, Baratz and Peleg [5]. An (α, β)-LAST is a rooted tree that has (a) length at most β times the MST and (b) the distance from any vertex to the root in the tree is at most α times the distance in the original metric. Khuller et al. [18] showed that every metric has an $\left(\alpha, 1 + \frac{2}{\alpha - 1}\right)$-LAST (for any $\alpha > 1$) and this is best possible.

2 Model and Preliminaries

The input to the *Heterogenous TSP* (HetTSP) consists of a metric (V, d) denoting distances between vertices, a depot $r \in V$ and k vehicles with speeds $\{\lambda_i\}_{i=1}^k$ greater than or equal to 1. The goal is to find tours $\{\tau_i\}_{i=1}^k$ (starting and ending at r) for each vehicle so that every vertex is covered in some tour and the *maximum completion time* $\max_{i=1}^k \frac{d(\tau_i)}{\lambda_i}$ is minimized.

At the loss of a factor of two in the approximation, we assume that the λ_i's are all (non-negative integral) powers of 2. Then, for each integer $i \geq 0$ we use μ_i to denote the number of vehicles with speed 2^i. Let OPT denote the optimal value of this modified instance of HetTSP.

Let $G = (V, E)$ be the complete graph on vertices V with edge-weights corresponding to the distance function d. For any set $F \subseteq E$ of edges, we set $d(F) = \sum_{e \in F} d_e$. Given any (multi)graph H and a subset U of its vertices, $H[U]$ denotes the subgraph induced on U and H/U denotes the graph obtained

by contracting vertices U to a single vertex (we retain parallel edges). Moreover, for any pair of vertices u, v in H, we use $d_H(u, v)$ to denote the length of the shortest path in H between u and v.

3 Algorithm for HetTSP

Assume that we have correctly guessed a value M such that $\frac{M}{2} \leq \mathsf{OPT} \leq M$ (we address how to find M in the end of Sect. 3.3.) We partition the set of vertices V according to their distance to r:

$$V_0 = \{u \in V : d(r, u) \leq M\}, \text{ and}$$

$$V_i = \{u \in V : d(r, u) \in (2^{i-1}M, 2^i M]\}, \text{ for all } i \geq 1.$$

The vertices in V_i are referred to as *level* i vertices. For any $i \geq 0$, we use $V_{\leq i}$ as a shorthand for $\cup_{j=0}^{i} V_j$ and similarly $V_{<i} = \cup_{j=0}^{i-1} V_j = V_{\leq i-1}$.

We define the *level of an edge* $(u, v) \in E$ as the larger of the levels of u and v. For each $i \geq 0$, E_i denotes the edges in E of level i. Note that $d_e \leq 2^{i+1}M$ for all $e \in E_i$, since both end-points of e are in $V_{\leq i}$ and the triangle inequality bounds its distance by the two-hop path via the root. We use the notation $E_{\leq i} = \cup_{j=0}^{i} E_j$ and $E_{\geq i} = \cup_{j \geq i} E_j$.

3.1 Assignable Trees

We start by studying collections of trees that can be assigned to vehicles in a way that each vehicle takes time $O(M)$ to visit all of its assigned trees. In the following let α and β be integers.

Definition 1 (Assignable Trees). *A collection of r-rooted trees $\bigcup_{i \geq 0} \mathcal{T}_i$ covering all vertices V is called* (α, β)-assignable *if it satisfies the following properties.*

1. *For each $i \geq 0$ and every $T \in \mathcal{T}_i$, $d(T) \leq \alpha\, 2^i\, M$.*
2. *For each $i \geq 0$, $\sum_{j \geq i} d(\mathcal{T}_j) \leq \beta M \sum_{j \geq i-1} 2^j\, \mu_j$.*

Intuitively, the trees in $\mathcal{T}_i$ can be assigned to vehicles with speed 2^i so as to complete in time $O(\alpha M)$. Condition (2) guarantees that the trees $\bigcup_{j \geq i} \mathcal{T}_i$ targeted by vehicles of speed 2^{i-1} and above stand a chance of being handled by them within makespan $O(\beta M)$. Interestingly, these minimal conditions are enough to eventually assign all trees in the collection to vehicles while guaranteeing makespan $O((\alpha + \beta)M)$.

Lemma 1. *Given an assignable collection $\bigcup_{i \geq 0} \mathcal{T}_i$ of r-rooted trees, we can obtain in polynomial time an $(4\alpha + 2\beta)$-approximation for* HetTSP.

To prove this lemma[2], we show that condition (2) guarantees the existence of a fractional assignment of trees where each vehicle incurs load at most βM.

[2] We remark that a direct proof of Lemma 1 is also possible, but the route we take reveals more properties of the requirement at hand and could potentially be useful in tackling generalizations of HetTSP.

Then using condition (1) and a result on scheduling on parallel machines [19], we round this assignment into an integral one while increasing the load on each vehicle by at most $2\alpha M$. We lose an extra factor of 2 to convert the trees into routes.

Fractional Assignment. Consider the bipartite graph H whose left side contains one node for each tree in $\bigcup_i \mathcal{T}_i$ and whose right side contains one node for each vehicle. (We identify the nodes with their respective trees/vehicles.) There is an arc between the tree $T \in \mathcal{T}_i$ and a vehicle of speed 2^j if $j \geq i-1$.

Consider the following b-matching problem in H: for each tree $T \in \mathcal{T}_i$, we set $b(T) = d(T)$ and for each vehicle u of speed 2^j we set $b(u) = \beta 2^j M$. A (left-saturating) b-matching is one which fractionally assigns all $b(T)$ units of each tree T such that no vehicle u is assigned more than $b(u)$ units. Notice that a feasible b-matching gives a fractional assignment of trees where each vehicle j of speed 2^j incurs fractional makespan at most βM.

Then our goal is to show the existence of a b-matching in H. Using a standard generalization of Hall's Theorem (e.g., see page 54 of [11]), we see that H has a feasible b-matching iff for every set V of trees, $\sum_{T \in V} b(T)$ is at most $\sum_{u \in N(V)} b(u)$, where $N(V)$ is the neighborhood of V. However, the structure of H allows us to focus only on sets V which are equal to $\bigcup_{j \geq i} \mathcal{T}_j$ for some i.[3] Using this revised condition, H has a b-matching iff for all i, $\sum_{j \geq i} d(\mathcal{T}_j) \leq \beta M \sum_{j \geq i-1} 2^j \mu_j$. Since this is exactly condition (2) in Definition 1, it follows that H indeed has a b-matching (which can be obtained in polynomial time using any maximum flow algorithm [11]).

Scheduling Parallel Machines. We show how to round the fractional assignment obtained in the previous section. We consider each tree as a "job" and each vehicle as a "machine", where the "processing time" $p_{T,u}$ of a tree T in a vehicle u of speed 2^j is $d(T)/2^j$; then the "makespan" of a vehicle is exactly equal to the sum of the processing times of the trees assigned to it.

Let $x_{T,u}$ denote the fraction of tree T assigned to vehicle u given by scaling down a b-matching in H (i.e., if the matching assigns d units of T to vehicle u, we have $x_{T,u} = d/d(T)$). The feasibility of the matching gives $\sum_T x_{T,u} p_{T,u} \leq \beta M$ for all u. Moreover, by construction of the edges of H, $x_{T,u} > 0$ for $T \in \mathcal{T}_i$ implies that u has speed at least 2^{i-1}. Then using property (1) of assignable trees we get that $x_{T,u} > 0$ implies $p_{T,u} \leq 2\alpha M$. These two properties guarantee that x is a feasible solution for the natural LP formulation for the scheduling problem with a feasible makespan value of βM and the maximum processing time t set to $2\alpha M$. Theorem 1 of [19] then asserts that x can be rounded into an integral assignment of trees to vehicles such that the load on any vehicle is at most $(2\alpha + \beta)M$.

[3] To see that all other inequalities are dominated by those coming from such sets, first notice that if V contains a tree in $\mathcal{T}_i$ then $N(V)$ already contains all vehicles of speed 2^j for $j \geq i-1$. Then adding to V extra trees in $\bigcup_{j \geq i} \mathcal{T}_j$ does not change its neighborhood and thus leads to a dominating inequality.

As in Sect. 1.1, we can transform each tree in $\bigcup_{i\geq 0} \mathcal{T}_i$ into a cycle while at most doubling its length, which then gives a $(4\alpha+2\beta)$-approximation for HetTSP. This concludes the proof of Lemma 1.

3.2 Level-Prim Spanning Tree

In order to obtain an assignable collection of r-rooted trees for our instance, we formally introduce Level-Prim trees. These are the trees obtained by the following procedure.

Algorithm 1. Level-Prim(G)

1: For each $i \geq 0$, let H_i be an MST for $G[V_{\leq i}]/V_{<i}$.
2: **return** $\mathcal{H} = \bigcup_{i\geq 0} H_i$.

Note that Level-Prim trees can alternately be defined by modifying Prim's algorithm such that nodes in level i are only considered to be added to the tree after all nodes in levels below i have already been added.

Theorem 3. *A* Level-Prim *tree* $\mathcal{H} = \{H_i\}_{i\geq 0}$ *satisfies the following:*

- *The vertex-levels along every root-leaf path are non-decreasing.*
- *For each* $i \geq 0$, $\sum_{j\geq i} d(H_j) \leq 8 \cdot \mathsf{MST}\,(G/V_{<i})$.

Before we prove Theorem 3, we note that the second property in the theorem mirrors the second property in Definition 1. A formal connection between the two is established via the following lemma that uses an optimal vehicle routing solution to derive a feasible spanning tree connecting a suffix of the level sets.

Lemma 2 (Lower Bound). *For each level* $\ell \geq 0$, $\mathsf{MST}(G/V_{<\ell}) \leq M \cdot \sum_{j\geq \ell-1} 2^j \mu_j$.

Proof. Consider an optimal solution for HetTSP and let E^* be the set of edges traversed by vehicles in this solution; label each edge in E^* by the vehicle that traversed it. Clearly E^* connects all vertices to the root r.

Only vehicles having speed at least $2^{\ell-1}$ can reach any vertex in $V_{\geq \ell}$ (since a vehicle of speed s travels distance at most $s \cdot \mathsf{OPT} \leq s \cdot M$). Thus every edge in $E^* \cap E_{\geq \ell}$ must be labeled by some vehicle of speed at least $2^{\ell-1}$. This implies that $d\,(E^* \cap E_{\geq \ell}) \leq M \cdot \sum_{j\geq \ell-1} 2^j\, \mu_j$, since the right hand side is a bound on the total length traversed by vehicles having speed at least $2^{\ell-1}$.

On the other hand, since E^* connects all vertices, $E^* \cap E_{\geq \ell}$ contains a spanning tree of $G/V_{<\ell}$. Thus $\mathsf{MST}(G/V_{<\ell}) \leq d\,(E^* \cap E_{\geq \ell}) \leq M \cdot \sum_{j\geq \ell-1} 2^j\, \mu_j$.

We get the following corollary of Theorem 3.

Corollary 1. *A* Level-Prim *tree* $\mathcal{H} = \{H_i\}_{i\geq 0}$ *satisfies the following:*

- *The vertex-levels along every root-leaf path are non-decreasing.*
- *For each* $i \geq 0$, $\sum_{j\geq i} d(H_j) \leq 8M \sum_{j\geq i-1} 2^j\, \mu_j$.

In the rest of this section, we prove Theorem 3. It is easy to see that for every ℓ, $\bigcup_{j=1}^{\ell} H_j$ spans $G[V_{\leq \ell}]$, hence the procedure produces a spanning tree for G. Moreover, by construction we obtain that every root-leaf path in $\mathcal{H}$ traverses the levels in non-decreasing order as desired. Thus, we focus on proving the second property in the theorem.

Instead of comparing the length of the edges in $\mathcal{H}$ with an MST, it turns out to be much easier to use a specific LAST tree as a proxy for the latter. The following LAST is implicit in the construction given in [18] and is omitted. Recall that a *spider* is a tree with at most one vertex (the center) having degree greater than two.

Theorem 4 ([18]). *Given any metric (V, d) with root r, there exists a spanning spider L with center r such that:*

- *For each $u \in V$, the distance from r to u in L is at most $2 \cdot d(r, u)$.*
- *The length of L is at most four times the* MST *in (V, d), i.e. $d(L) \leq 4 \cdot \mathsf{MST}$.*

We remark that we cannot use a LAST directly instead of Level-Prim since the former does not need to have the properties asserted by Theorem 3. Using these spider LASTs we can obtain the main lemma needed to complete the proof of Theorem 3.

Lemma 3. *For any graph G and any* Level-Prim *tree $\mathcal{H}$ on G, we have $d(\mathcal{H}) \leq 8 \cdot \mathsf{MST}(G)$.*

Proof. Consider a spider LAST L for G from Theorem 4, and let $\mathcal{P}$ denote the set of all root-leaf paths in L; note that the paths in $\mathcal{P}$ are edge-disjoint.

Consider any root-leaf path $P = (r = u_1 \to u_2 \to \ldots \to u_k)$ in $\mathcal{P}$. We claim that P crosses levels almost in an increasing order. Specifically, there does not exist a pair of nodes $u_i, u_j \in P$ with $i < j$, $u_i \in V_\ell$ and $u_j \in V_{\leq \ell-2}$. Suppose (for a contradiction) that this were the case; then we would have that

$$d_L(r, u_j) = d_L(r, u_i) + d_L(u_i, u_j) \geq d_L(r, u_i) \geq d(r, u_i) > 2^{\ell-1} M,$$

where the last inequality uses $u_i \in V_\ell$. On the other hand, $d(r, u_j) \leq 2^{\ell-2} M$ since $u_j \in V_{\leq \ell-2}$; so we obtain $d_L(r, u_j) > 2 \cdot d(r, u_j)$, which contradicts the definition of L (see Theorem 4).

Now we transform L into another spider L' which traverses levels in non-decreasing order as follows. For each root-leaf path $P = (r = u_1 \to u_2 \to \ldots \to u_k)$, perform the following modification. Let $\{a_1, a_2, \ldots, a_{k'}\}$ be the subsequence of P consisting of the vertices in *even numbered* levels, i.e. each $a_i \in L_{2\ell}$ for some $\ell \geq 0$. Similarly, let $\{b_1, b_2, \ldots, b_{k''}\}$ be the subsequence of P consisting of the vertices in *odd numbered* levels. Define two paths $P_{even} := (r \to a_1 \to \ldots \to a_{k'})$ (shortcutting P over nodes b_i's) and $P_{odd} = (r \to b_1 \to \ldots \to b_{k''})$ (shortcutting P over a_i's). Observe that both P_{even} and P_{odd} cross levels *monotonically*: if not then there must be some $i < j$ in P with $u_i \in V_\ell$ and $u_j \in V_{\leq \ell-2}$, contrary to the previous claim. Also, by employing the triangle inequality we have that $d(P_{even})$, $d(P_{odd}) \leq d(P)$. Finally define the spider L' as the union of the paths $\{P_{even}, P_{odd}\}$ over all root-leaf paths $P \in \mathcal{P}$.

By construction, the vertex levels along each root-leaf path of L' are non-decreasing. Additionally $d(L') = \sum_{P \in \mathcal{P}} (d(P_{even}) + d(P_{odd})) \leq 2 \sum_{P \in \mathcal{P}} d(P) = 2 \cdot d(L) \leq 8 \cdot \mathsf{MST}$, by Theorem 4. Now partition the edges of L' as:

$$\Delta_\ell = \begin{cases} L'[V_0] & \text{if } \ell = 0, \\ L'[V_{\leq \ell}] \setminus L'[V_{\leq \ell - 1}] & \text{if } \ell \geq 1. \end{cases}$$

By the monotone property of paths in L', it follows that $L'[V_{\leq \ell}]$ is connected for every $\ell \geq 0$. Thus Δ_ℓ is a *spanning tree* in graph $G[V_{\leq \ell}]/V_{<\ell}$. Since H_ℓ in the Level-Prim construction, is chosen to be an MST in $G[V_{\leq \ell}]/V_{<\ell}$, we obtain $d(H_\ell) \leq d(\Delta_\ell)$. So, $d(\mathcal{H}) = \sum_{\ell \geq 0} d(H_\ell) \leq \sum_{\ell \geq 0} \Delta_\ell = d(L') \leq 8 \cdot \mathsf{MST}$. This completes the proof of the lemma.

Completing proof of Theorem 3. We now prove the second property in Theorem 3. Lemma 3 directly implies this property for $i = 0$. For any level $i > 0$ consider the graph $G' = G/V_{<i}$; observe that $\bigcup_{j \geq i} H_j$ is a Level-Prim tree for G' (due to the iterative construction of $\mathcal{H} = \bigcup_{\ell \geq 0} H_\ell$). Thus applying Lemma 3 to graph G' and its Level-Prim tree $\bigcup_{j \geq i} H_j$, we have $\sum_{j \geq i} d(H_j) \leq 8 \cdot \mathsf{MST}(G') = 8 \cdot \mathsf{MST}(G/V_{<i})$.

3.3 Decomposition Procedure

In this section we decompose a Level-Prim tree into an assignable collection $\bigcup_{i \geq 0} \mathcal{T}_i$ of r-rooted trees. Motivated by Corollary 1, the idea is to break each subgraph H_i into many pieces and connect them to r in order to form the set of trees $\mathcal{T}_i$. Suppose that each connected component in H_i is large enough, i.e. has length at least $2^i M$. Then for each $i \geq 0$, break the connected components of H_i into trees of length approximately $2^i M$; add to each tree the shortest edge connecting them to r and set $\mathcal{T}_i$ as the collection of r-rooted trees obtained. By construction we get that $\bigcup_{i \geq 0} \mathcal{T}_i$ satisfies the first property of an assignable collection. Moreover, notice that each edge added to connect a tree to the root has approximately the same length as the tree itself; this guarantees that $d(\mathcal{T}_i) \lesssim 2d(H_i)$. It then follows that the collection $\bigcup_{i \geq 0} \mathcal{T}_i$ is assignable.

Notice that it was crucial to break H_i into trees of length at least approximately 2^i. But this is problematic when H_i has a small connected component. In this case we show that such a small component is always attached to ("dangling" from) a large enough component in H_{i-1} (otherwise the dangling edge to a much earlier level will already make this component heavy enough not to be small); then we simply treat the small component as part of the latter in the assignment.

Now we formally describe the proposed decomposition procedure.

Step 1. Let $\mathcal{S}_0$ contain the subtree $H_0 = \mathcal{H} \bigcap E_0$. For each level $i \geq 1$: partition edges $\mathcal{H} \bigcap E_i$ into a collection $\mathcal{S}_i$ of (unrooted) subtrees such that each subtree contains exactly one edge from $V_{<i}$ to V_i. For any $\tau \in \mathcal{S}_i$ call the unique edge from $V_{<i}$ to V_i its *head-edge* $h(\tau)$. Note that such a partition is indeed possible since $\mathcal{H}[V_{\leq i}]/V_{<i}$ is connected.

Any subtree in $\mathcal{S}_i$ (for $i \geq 0$) is referred to as a level i subtree. Note that head-edges are defined only for subtrees in level 1 and above.

Step 2. For each level $i \geq 0$: *mark* those $\tau \in \mathcal{S}_i$ that have $d(\tau) \geq 2^{i-3}M$. In addition, *mark* the tree H_0 in $\mathcal{S}_0$. Let $\mathcal{S}_i^m$ and $\mathcal{S}_i^u$ denote the marked and unmarked subtrees in $\mathcal{S}_i$.

Step 3. For each level $i \geq 1$ and unmarked $\sigma \in \mathcal{S}_i^u$: define $\pi(\sigma)$ as the subtree in $\bigcup_{j<i} \mathcal{S}_j$ containing the other end-point of $h(\sigma)$.

Claim. For $i \geq 1$ and unmarked $\sigma \in \mathcal{S}_i^u$, $\pi(\sigma) \in \mathcal{S}_{i-1}$. Moreover, $\pi(\sigma)$ is marked.

Proof. Since σ is unmarked in level $i \geq 1$, $d(h(\sigma)) \leq d(\sigma) < 2^{i-3}M$. So the end-point v of $h(\sigma)$ in $\pi(\sigma)$ satisfies $d(r,v) \geq \frac{3}{2} \cdot 2^{i-2}M$, otherwise $d(h(\sigma)) \geq 2^{i-1}M - d(r,v) > 2^{i-3}M$. In particular $v \in V_{i-1}$ and thus $\pi(\sigma) \in \mathcal{S}_{i-1}$.

For the second part of the claim, notice that if $i = 1$ then $\pi(\sigma) = H_0$, which is always marked. So suppose $i \geq 2$. From the above, $\pi(\sigma)$ is in level $i - 1 \geq 1$ and hence contains a head-edge. This implies that $\pi(\sigma)$ contains some vertex $u \in V_{<i-1}$, namely an end-point of $h(\pi(\sigma))$. But then $d(\pi(\sigma)) \geq d(u,v) \geq d(r,v) - d(r,u) \geq 2^{i-3}M$, where we used $d(r,u) \leq 2^{i-2}M$ since $u \in V_{<i-1}$ and $d(r,v) \geq \frac{3}{2} \cdot 2^{i-2}M$ from above. Thus $\pi(\sigma)$ must be marked.

Step 4. For each level $i \geq 0$ and marked $\tau \in \mathcal{S}_i^m$: define $\mathsf{Dangle}(\tau) = \pi^{-1}(\tau)$ as the set of all unmarked $\sigma \in \mathcal{S}_{i+1}^u$ having $\pi(\sigma) = \tau$. Clearly $d(\sigma) \leq 2^{i-2}M$ for all $\sigma \in \mathsf{Dangle}(\tau)$.

Step 5. For each level $i \geq 0$ and marked $\tau \in \mathcal{S}_i^m$: partition the tree $\tau \cup \mathsf{Dangle}(\tau)$ into subtrees $T_1, \ldots, T_q$ such that the first $q-1$ trees have length in the range $[2^{i+1}M, 2^{i+2}M]$ and T_q has length at most $2^{i+2}M$. Notice that this is possible since all edges of $\tau \cup \mathsf{Dangle}(\tau)$ belong to $E_{\leq i+1}$ and hence have length at most $2^{i+1}M$. Finally, add the shortest edge from r to each of these new subtrees to obtain a collection $\mathcal{T}_i(\tau)$ of r-rooted trees.

Claim. For any $T \in \mathcal{T}_i(\tau)$, we have $d(T) \leq 3 \cdot 2^{i+1}M$.

Proof. Notice that every $T \in \mathcal{T}_i(\tau)$ consists of a T_j (for some $1 \leq j \leq q$) and an edge from r to a node in $V_{\leq i+1}$. Since the former has length at most $2^{i+2}M$ and the latter has length at most $2^{i+1}M$, it follows that $d(T) \leq 3 \cdot 2^{i+1}M$.

Claim. $\sum_{T \in \mathcal{T}_i(\tau)} d(T) \leq 5 \cdot [d(\tau) + d(\mathsf{Dangle}(\tau))]$.

Proof. We break the analysis into two cases depending of q. Suppose $q = 1$, namely $\mathcal{T}_i(\tau)$ consists of a single tree T. In this case $T = \tau \cup \mathsf{Dangle}(\tau) \cup \{\mathsf{e}\}$, where e is an edge to r. If $i = 0$ then $d(e) = 0$ and the result holds directly. If $i > 0$ then τ has a node in $V_{<i}$ and hence $d(e) \leq 2^{i-1}M$. Because τ is marked and different than H_0, the lower bound on its length implies that $d(e) \leq 2^{i-1}M \leq 4d(\tau) \leq 4(d(\tau) + d(\mathsf{Dangle}(\tau))$. The result follows by adding the length of $\tau \cup \mathsf{Dangle}(\tau)$ to both sides.

Now suppose $q > 1$. Since all trees in $\mathcal{T}_i(\tau)$ lie in $V_{\leq i+1}$, each edge from the root in $\mathcal{T}_i(\tau)$ has length at most $2^{i+1}M$. So the left hand side is at most $\sum_{j=1}^{q} d(T_j) + q \cdot 2^{i+1}M$. But for $j < q$ we have $d(T_j) \geq 2^{i+1}M$, so the last term of

the previous expression can be upper bounded by $\frac{q}{(q-1)} \sum_{j=1}^{q-1} d(T_j)$. This bound is smallest when $q = 2$, which then gives $\sum_{T \in \mathcal{T}_i(\tau)} d(T) \leq 3 \sum_{j=1}^{q} d(T_j) \leq 3d(\tau)$. This concludes the proof of the claim.

Step 6. For each level $i \geq 0$: define $\mathcal{T}_i = \bigcup_{\tau \in \mathcal{S}_i^m} \mathcal{T}_i(\tau)$.

The following lemma summarizes the main property of our decomposition procedure.

Lemma 4. *The collection* $\{\mathcal{T}_i\}_{i \geq 0}$ *obtained from the above procedure is* $(6, 40)$*-assignable.*

Proof. By the first claim above, each tree in $\mathcal{T}_i$ has length at most $3 \cdot 2^{i+1} M$. So the collection satisfies condition (1) of Definition 1.

Fix any $i \geq 0$ for condition (2) in Definition 1. Due to Corollary 1, it suffices to prove that $\sum_{j \geq i} d(\mathcal{T}_j) \leq 5 \cdot \sum_{j \geq i} d(H_j)$. Using the second claim from above, we obtain that

$$d(\mathcal{T}_j) = \sum_{\tau \in \mathcal{S}_j^m} d(\mathcal{T}_i(\tau)) \leq 5 \cdot \sum_{\tau \in \mathcal{S}_j^m} [d(\tau) + d(\mathsf{Dangle}(\tau))] = 5 \cdot d(\mathcal{S}_j^m) + 5 \cdot d(\mathcal{S}_{j+1}^u).$$

The last equality above uses the fact that that $\{\mathsf{Dangle}(\tau) : \tau \in \mathcal{S}_j^m\}$ is a partition of $\mathcal{S}_{j+1}^u$. Thus:

$$\sum_{j \geq i} d(\mathcal{T}_j) \leq 5 \cdot \sum_{j \geq i} d(\mathcal{S}_j^m) + 5 \cdot \sum_{j \geq i} d(\mathcal{S}_{j+1}^u) \leq 5 \cdot \sum_{j \geq i} d(\mathcal{S}_j) = 5 \cdot \sum_{j \geq i} d(H_j). \quad (1)$$

This concludes the proof of Lemma 4.

Summary of the Algorithm. Our algorithm starts with an initial low guess of M and runs the Level-Prim procedure. If the second condition in Corollary 1 does not hold for this run, we double the guess for M and repeat until it is satisfied (this happens the first time that M reaches the condition for the correct guess: $\frac{M}{2} \leq \mathsf{OPT} \leq M$). We use the decomposition in this section summarized in Lemma 4 to obtain a (6,40)-assignable collection of trees. Using Lemma 1 on this collection gives us the desired constant approximation ratio by observing that the guess M in this step obeys $M \leq 2 \cdot \mathsf{OPT}$.

4 Open Problems

One interesting open question regards the approximability of HetTSP and HetCVRP when vehicles are located in multiple different depots across the space. The current definition of an assignable collection and the definition of Level-Prim crucially depend on the assumption of a unique depot, hence an extension to the multi-depot case is likely to require new ideas. Another interesting direction is to consider HetCVRP with non-uniform capacities, where our metric reduction ideas do not seem to generalize directly.

References

1. Altinkemer, K., Gavish, B.: Heuristics for unequal weight delivery problems with a fixed error guarantee. Operations Research Letters 6, 149–158 (1987)
2. Altinkemer, K., Gavish, B.: Heuristics for delivery problems with constant error guarantees. Transportation Research 24, 294–297 (1990)
3. Arkin, E.M., Hassin, R., Levin, A.: Approximations for minimum and min-max vehicle routing problems. Journal of Algorithms 59(1), 1–18 (2006)
4. Arora, S.: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. J. ACM 45, 753–782 (1998)
5. Awerbuch, B., Baratz, A., Peleg, D.: Cost-sensitive analysis of communication protocols. In: Proceedings of the 9th Annual Symposium on Principles of Distributed Computing, pp. 177–187 (1990)
6. Blum, A., Chawla, S., Karger, D.R., Lane, T., Meyerson, A., Minkoff, M.: Approximation algorithms for orienteering and discounted-reward tsp. SIAM J. Comput. 37(2), 653–670 (2007)
7. Bompadre, A., Dror, M., Orlin, J.: Probabilistic Analysis of Unit Demand Vehicle Routing Problems. J. Appl. Probab. 44, 259–278 (2007)
8. Chekuri, C., Korula, N., Pál, M.: Improved algorithms for orienteering and related problems. In: Teng, S.-H. (ed.) SODA, pp. 661–670. SIAM, Philadelphia (2008)
9. Chekuri, C., Kumar, A.: Maximum coverage problem with group budget constraints and applications. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) RANDOM 2004 and APPROX 2004. LNCS, vol. 3122, pp. 72–83. Springer, Heidelberg (2004)
10. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Report 388, Graduate School of Industrial Administration, CMU (1976)
11. Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A.: Combinatorial optimization. John Wiley & Sons, Inc., New York (1998)
12. Das, A., Mathieu, C.: A Quasi-polynomial Time Approximation Scheme for Euclidean Capacitated Vehicle Routing. In: Charikar, M. (ed.) SODA, pp. 390–403. SIAM, Philadelphia (2010)
13. Even, G., Garg, N., Könemann, J., Ravi, R., Sinha, A.: Min-max tree covers of graphs. Oper. Res. Lett. 32(4), 309–315 (2004)
14. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. SIAM J. Comput. 7(2), 178–193 (1978)
15. Garg, N.: Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In: Proceedings of the 34th Annual ACM Symposium on the Theory of Computing, pp. 396–402 (2005)
16. Gupta, A., Nagarajan, V., Ravi, R.: Approximation Algorithms for VRP with Stochastic Demands (2010) (submitted)
17. Haimovich, M., Kan, A.H.G.R.: Bounds and heuristics for capacitated routing problems. Mathematics of Operations Research 10(4), 527–542 (1985)
18. Khuller, S., Raghavachari, B., Young, N.E.: Balancing minimum spanning trees and shortest-path trees. Algorithmica 14(4), 305–321 (1995)
19. Lenstra, J.K., Shmoys, D.B., Tardos, E.: Approximation algorithms for scheduling unrelated parallel machines. Mathematical Programming 46, 259–271 (1990), doi:10.1007/BF01585745
20. Mitchell, J.S.B.: Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric TSP, k-MST, and Related Problems. SIAM Journal on Computing 28(4), 1298–1309 (1999)
21. Toth, P., Vigo, D. (eds.): The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA, USA (2002)

Approximation Algorithms for Single and Multi-Commodity Connected Facility Location

Fabrizio Grandoni[1] and Thomas Rothvoß[2]

[1] University of Rome Tor Vergata, Italy
grandoni@disp.uniroma2.it
[2] MIT, USA
rothvoss@math.mit.edu

Abstract. In the classical *facility location* problem we are given a set of facilities, with associated opening costs, and a set of clients. The goal is to open a subset of facilities, and to connect each client to the closest open facility, so that the total connection and opening cost is minimized. In some applications, however, open facilities need to be connected via an infrastructure. Furthermore, connecting two facilities among them is typically more expensive than connecting a client to a facility (for a given path length). This scenario motivated the study of the *connected facility location* problem (CFL). Here we are also given a parameter $M \geq 1$. A feasible solution consists of a subset of open facilities and a Steiner tree connecting them. The cost of the solution is now the opening cost, plus the connection cost, plus M times the cost of the Steiner tree.

In this paper we investigate the approximability of CFL and related problems. More precisely, we achieve the following results:

• We present a new, simple 3.19 approximation algorithm for CFL. The previous best approximation factor is 3.92 [Eisenbrand, Grandoni, Rothvoß, Schäfer-'10].
• We show that SROB, i.e. the special case of CFL where all opening costs are 0, is hard to approximate within 1.28. The previous best lower bound for SROB is 1.01, and derives trivially from Steiner tree inapproximability [Chlebík, Chlebíková-'08]. The same inapproximability result extends to other well-studied problems, such as *virtual private network* and *single-sink buy-at-bulk*.
• We introduce and study a natural multi-commodity generalization MCFL of CFL. In MCFL we are given source-sink pairs (rather than clients) that we wish to connect. A feasible solution consists of a subset of open facilities, and a forest (rather than a tree) spanning them. Source-sink connection paths can use several trees in the forest, but must enter and leave each tree at open facilities. We present the first constant approximation for MCFL.

1 Introduction

In the classical *metric facility location* problem (FL), we are given an undirected graph $G = (V, E)$, with edge costs (or weights) $c : E \to \mathbb{Q}^+$, a set of clients $C \subseteq V$, and a set of facilities $F \subseteq V$, with opening costs $o : F \to \mathbb{Q}^+$. A feasible solution is given by a subset $F' \subseteq F$ of *open* facilities. The goal is to minimize the opening

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 248–260, 2011.

cost of F', plus the shortest path distance from each client to the closest open facility. More formally, let $c(v,u)$ denote the shortest path distance between u and v, and $c(v,U) := \min_{u \in U} c(v,u)$ for any $U \subseteq V$. Then the objective function to be minimized is $\sum_{f \in F'} o(f) + \sum_{v \in C} c(v,F')$.

In several applications one needs to connect open facilities via an infrastructure. Typically, connecting facilities among them is more expensive than connecting clients to facilities. This scenario motivated the introduction of the following problem. Let $c(E') := \sum_{e \in E'} c(e)$ for any $E' \subseteq E$, and $c(G') = c(E(G'))$ for any subgraph G' of G.

> CONNECTED FACILITY LOCATION (CFL). Given an undirected graph $G = (V,E)$, with edge costs $c : E \to \mathbb{Q}^+$, a set of clients $C \subseteq V$, a set of facilities $F \subseteq V$, with opening costs $o : F \to \mathbb{Q}^+$, and a parameter $M \geq 1$. Compute a subset $F' \subseteq F$ of open facilities, and a tree T' spanning F', in order to minimize $\sum_{f \in F'} o(f) + \sum_{v \in C} c(v,F') + M \cdot c(T')$.

CFL is well-studied in the literature [9,18,21,28]. The current best approximation for it is 3.92 [9]. A very well-studied [9,18,20,23,28] special case of CFL is the *single-sink rent-or-buy* problem (SROB), where $F = V$ and opening costs are zero. In this context, we can think of edges of T' as *bought* edges (for which we pay a fixed, large cost), and edges outside T' as *rented* edges (for which we pay a cost proportional to the number of paths using them).

Another way to interpret CFL is as follows. Clients are users who want to reach a public transportation network T' each day to get to their office. Commuting has a social cost which is shared on T', and payed on an individual basis outside T'. Here open facilities are stations at which users can access T'. This view of CFL suggests a natural multi-commodity generalization of the problem. Replace clients with origin-destination pairs, and imagine that you can construct several, possibly disconnected, transportation networks. Each network can be reached and left at stations. More formally, one can define the following problem. For a forest T' and a subset of nodes V', let $c_{V',T'}(u,v)$ be the shortest path distance between nodes u and v after adding one edge of cost zero between each pair of nodes in V' belonging to the same tree of T'.

> MULTI-COMMODITY CONNECTED FACILITY LOCATION (MCFL). Given an undirected graph $G = (V,E)$, with edge costs $c : E \to \mathbb{Q}^+$, a set of source-sink pairs[1] $P = \{(s_1,r_1),\ldots,(s_k,r_k)\}$, $s_i, r_i \in V$, a set of facilities $F \subseteq V$, with opening costs $o : F \to \mathbb{Q}^+$, and a parameter $M \geq 1$. Compute a subset $F' \subseteq F$ of open facilities, and a forest T', in order to minimize $\sum_{f \in F'} o(f) + \sum_{(s,r) \in P} c_{F',T'}(s,r) + M \cdot c(T')$.

To the best of our knowledge, MCFL was not addressed before (at least, from the point of view of approximation algorithms). However, there is a special case of the problem which is well-studied in the literature: the *multi-commodity rent-or-buy* problem (MROB) is the special case of MCFL where $F = V$ and opening costs are zero [2,3,12,20,24]. A solution to an MROB instance consists of a forest T' of bought edges.

[1] For notational convenience, we will consider P as a multi-set of pairs.

The cost of the solution is given by $M \cdot c(T') + \sum_{(s,r)\in P} c_{T'}(s,r)$, where $c_{T'}(u,v)$ denotes the shortest path distance between u and v, after contracting the connected components of T'. In other terms, MROB is the multi-commodity version of SROB.

1.1 Our Results and Techniques

In this paper we study the approximability of CFL and related problems. In particular, we obtain the following three main results.

(1) An Improved Approximation for CFL. We present a 3.19 approximation algorithm for CFL, improving on the previous best 3.92 approximation [9]. The approximation algorithms for CFL by Gupta, Srinivasan and Tardos [21] and Eisenbrand, Grandoni, Rothvoß, and Schäfer [9] are both based on simple random sampling steps (more details in Section 1.2). Here we present a third, simple random sampling algorithm. We first randomly sample clients, and buy a Steiner tree T over them. Then we define a facility location instance (on all the clients), where the opening cost of each facility f is increased by the cost of augmenting T to include f. This way, the modified opening cost encodes both the real opening cost, and the cost of connecting f to the other open facilities via T.

Like in [9], our technique can be extended to the connected version of some variants of facility location. However, our approach is more flexible. For example, differently from [9], it gives a constant approximation for connected facility location with hard capacities. Due to space limits, extensions will be discussed in the full version of the paper.

From the analytical point of view, we exploit the *core-detouring* technique in [9], which was already successfully applied in the analysis of CFL [9] and related problems [15]. The basic idea is bounding the cost of connecting a set of clients to a random subset of them. This bound is based on detouring connection paths through a proper connected *core* graph. We cannot directly apply (as a black box) the core-detouring theorem in [9], since we need to connect clients to facilities rather than clients among them. However, the connection scheme used in the proof of the theorem has some particular properties that we can exploit for our purposes.

(2) A Constant Approximation for MCFL. We present the first constant approximation for MCFL: the approximation factor is 16.2. Our result is based on two main ingredients. The first ingredient is a reduction to the *prize-collecting* version of facility location, where we are allowed not to connect all the clients, but we have to pay a penalty for each disconnected client. More formally:

> PRIZE-COLLECTING FACILITY LOCATION (PFL). Given an undirected graph $G = (V, E)$, with edge costs $c : E \to \mathbb{Q}^+$, a set of clients $C \subseteq V$, with penalties $p : C \to \mathbb{Q}^+$, and a set of facilities $F \subseteq V$, with opening costs $o : F \to \mathbb{Q}^+$. Compute a subset $F' \subseteq F$ of open facilities and a subset $C' \subseteq C$ of *disconnected* clients, in order to minimize $\sum_{f\in F'} o(f) + \sum_{v\in C-C'} c(v, F') + \sum_{v\in C'} p(v)$.

In a MCFL solution there might be pairs which are connected directly via a shortest path (without using edges of the forest T'): intuitively, prizes are used to get rid of those pairs. In particular, each source and sink will define a client, and each pair containing at least one disconnected client in the PFL solution will be connected directly via a shortest path.

The second ingredient is a reduction to MROB. For each residual pair, we consider the associated pair of facilities in the PFL solution. This defines an MROB instance. On this instance, we run the MROB algorithm `rand` by Fleischer, Könemann, Leonardi, and Schäfer [12] (see also [20]). Here, we crucially exploit some properties of `rand` which are implicitly proved in [12]. In particular, using a different (possibly better) approximation algorithm for MROB might lead to a worse approximation for MCFL.

(3) A Stronger Inapproximability Result for SROB. Observe that CFL is not a generalization of FL, since its definition excludes $M = 0$. However, the techniques in [16] can be adapted to prove the same 1.463-inapproximability bound for CFL as for FL (we omit the proof for lack of space).

Theorem 1. *Unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log n)})$*, there is no polynomial time* 1.463*-approximation for CFL.*

Here we show that SROB, a very special case of CFL, is hard to approximate within 1.278. This greatly improves over the previously known approximation hardness of 1.01, which is based on the Steiner tree hardness result in [6] combined with a trivial reduction. The same hardness result extends immediately to other well-studied generalizations of SROB, as MROB and *single-sink buy-at-bulk* (SSBB). It also applies to *virtual private network* (VPN). (See Section 1.2 for omitted definitions). We remark that CFL is not a special case of the latter problems (hence, the results in [16] do not extend to them). Our result is based on a reduction to a special case of facility location, where facility costs are uniform, and client-facility distances are either 1 or 2. We show that the latter problem is hard via a reduction to a set-cover-like problem whose hardness was proved by Guha and Khuller [16].

1.2 Related Work

CFL is well-studied in the literature. Gupta, Kleinberg, Kumar, Rastogi, and Yener [18] obtain a 10.66-approximation for this problem, based on rounding an exponential size LP. Gupta, Srinivasan and Tardos [21] describe a random sampling algorithm for CFL, leading to a 9.01-approximation. Their algorithm randomly samples clients, and then runs an (unconnected) facility location approximation algorithm on the sampled clients: the corresponding open facilities form the set of open facilities in the final CFL solution. Swamy and Kumar [28] later improved the approximation to 8.55, using a primal-dual algorithm. The best-known result prior to our work is the 4.00-approximation by Eisenbrand, Grandoni, Rothvoß, and Schäfer [9]. They use a random-sampling approach subtly different from the one in [21]. In particular, they first solve an unconnected facility location problem on *all* the clients (not only on the sampled ones), and then randomly select a subset of the resulting (deterministic) pool of open facilities. Using the

improved Steiner tree approximation algorithm by Byrka, Grandoni, Rothvoß, and Sanità [5], the approximation factor reduces to 3.92. In this paper we present a third, still simple random-sampling algorithm for CFL.

A lot of research was devoted to a special case of CFL, namely SROB. The first constant approximation for SROB is given by Karger and Minkoff [23]. Gupta et al. [18] give a 9.01-approximation algorithm. Swamy and Kumar [28] describe a primal-dual 4.55-approximation algorithm for the same problem. Gupta, Kumar, Pal, and Roughgarden [20] propose a simple random sampling algorithm which gives a 3.55-approximation. Based on a refinement of the analysis in [20] via the core-detouring technique, the current best 2.80 approximation is given in [9].

Single-sink buy-at-bulk (SSBB) is the generalization of SROB, where we are given a set of cable types, each one with a cost and a capacity. Capacity on edges has to be reserved by installing zero or more copies of each cable type. The goal is sending one unit of flow from each source node to a sink. SROB can be seen as the special case of SSBB with two cable types: one of very small capacity and unit cost per unit capacity (corresponding to rented edges) and one of fixed cost and very large capacity (corresponding to bought edges). After a long sequence of improvements [13,14,17,20,22,26,29], the current best 20.41 approximation was recently given in [15].

The *virtual private network* problem (VPN), despite its rather different formulation, is intimately related to the other mentioned problems (see, e.g., [15]). Here, we are given upper and lower bounds on the amount of traffic that each node can send and receive. A solution is given by a capacity reservation and one path for each source-sink pair. The goal is minimizing the cost of the capacity reservation so that every traffic matrix which satisfies the upper bounds can be routed along the specified paths without exceeding edge capacities. Also this problem is well-studied [7,8,18,20]. The current best approximation is 2.80 [15].

As mentioned before, MCFL was not addressed before to the best of our knowledge. However, its special case MROB is a well-known problem. A $O(\log n)$-approximation for MROB is obtained by combining the approach by Awerbuch and Azar [2] with the refined Bartal trees in [11]. The first constant approximation is given by Kumar, Gupta, and Roughgarden [24] via the primal-dual method. A better and simpler random sampling algorithm is presented by Gupta et al. [20]. Based on a similar approach, the constant was later improved to 6.83 by Becchetti, Könemann, Leonardi, and Pál [3], and eventually to 5 by Fleischer, Könemann, Leonardi, and Schäfer [12].

In [24] the term MCFL is used to define a variant of MROB, where each connection path can use at most one tree in the forest T': let us call this problem 1-MROB. (A generalization of this problem, where subsets of pairs are grouped together, is discussed in [19]). The authors show that any β approximation for 1-MROB gives a 2β approximation for MROB, and present a constant approximation for the first problem. Indeed, essentially the same reduction works also in the opposite direction (giving a 10 approximation for 1-MROB based on the result in [12]). We can define the 1-MCFL problem analogously: this problem models natural scenarios (e.g., a person might want to use at most one public transportation network to commute). Using the same reduction as in [24], we obtain a $2 \cdot 16.2 = 32.4$ approximation for 1-MCFL: details are postponed to the full version of the paper.

Algorithm 1. Approximation algorithm for CFL

1. Guess a facility r from the optimum solution. Sample each client with probability $\frac{\alpha}{M}$. Let C' be the sampled clients.
2. Compute a ρ_{st}-approximate Steiner tree T on terminals $C' \cup \{r\}$.
3. Define a FL instance with clients C, facilities F and opening costs $o'(f) := o(f) + M \cdot c(f, C' \cup \{r\})$. Compute a (λ_F, λ_C)-approximate solution $F' \subseteq F$ to this instance.
4. Augment T with shortest paths from each $f \in F'$ to T. Let T' be the augmented tree.
5. Return (F', T')

2 An Improved Approximation for CFL

Let us consider Algorithm 1 in the figure. Here $\alpha \in (0, 1]$ is a constant to be fixed later. A *bifactor* (λ_F, λ_C)-approximation for FL is an algorithm which produces a solution to a FL instance of cost at most $\lambda_F \cdot O + \lambda_C \cdot C$, where O and C are the opening and connection cost of *any given* feasible solution. We will exploit the following result by Byrka [4].

Lemma 1. [4] *For any $\lambda_F > 1.67$, there is a $(\lambda_F, 1+2e^{-\lambda_F})$-approximation algorithm for FL.*

We remark that we would obtain an improved approximation also using a standard facility location algorithm. However, the approximation ratio would be slightly higher. A second tool that we need is the following Lemma which is implicitly proved in [9].

Lemma 2. [9] *Given an undirected graph $G = (V, E)$, with edge costs $c : E \to \mathbb{Q}^+$, a set of clients $C \subseteq V$, a subtree T' (core) containing a root node r, a mapping $\sigma : C \to V(T')$, and a probability $p \in (0, 1]$. Mark each client independently with probability p, and denote marked clients by C'. Let $\sigma(C') := \cup_{v \in C'} \sigma(v)$. Then $E[\sum_{v \in C} c(v, \sigma(C') \cup \{r\})] \leq \frac{0.807}{p} c(T') + \sum_{v \in C} c(v, \sigma(v))$.*

We let $OPT = (F^*, T^*)$ denote the optimal solution to the considered CFL instance, where F^* is the set of facilities and T^* the Steiner tree connecting them. We also let $\sigma^*(v) \in F^*$ be the facility serving $v \in C$ in OPT. By $OPT = O^* + C^* + S^*$ we denote the optimal cost, where $O^* := o(F^*)$ is the opening cost, $C^* := \sum_{v \in C} c(v, F^*)$ the connection cost, and $S^* := M \cdot c(T^*)$ the Steiner cost. Let $O_{fl} := \sum_{f \in F'} o'(f) = \sum_{f \in F'} (o(f) + M \cdot c(f, C' \cup \{r\}))$ be the opening cost of the facility location solution computed in Step 3, and $C_{fl} := \sum_{v \in C} c(v, F')$ be the connection cost in the same solution. We need a few, simple intermediate results.

Lemma 3. *The cost of the returned solution is at most $M \cdot c(T) + O_{fl} + C_{fl}$.*

Proof. The connection cost in the FL and CFL solutions are the same. Recall that $o'(f) = o(f) + M \cdot c(f, C' \cup \{r\})$. Thus the modified opening costs o' pay fully for both opening F' and for augmenting T to T'.

Lemma 4. *One has $E[M \cdot c(T)] \leq \rho_{st} \cdot (S^* + \alpha \cdot C^*)$.*

Proof. A feasible Steiner tree on $C' \cup \{r\}$, of expected cost $c(T^*) + \frac{\alpha}{M} C^*$, is obtained by augmenting T^* with the shortest paths between each $v \in C'$ and $\sigma^*(v)$. Multiplying by $\rho_{st} \cdot M$ then gives the claim.

Lemma 5. *One has* $E[O_{fl} + C_{fl}] \leq \lambda_F(O^* + \alpha C^*) + \lambda_C(C^* + \frac{0.807}{\alpha} S^*)$.

Proof. We provide a FL solution, whose expected opening cost is $O^* + \alpha\, C^*$ and whose expected connection cost is $C^* + \frac{0.807}{\alpha} S^*$. Choose facilities $\sigma^*(C') \cup \{r\}$, with $\sigma^*(C') := \cup_{v \in C'} \sigma^*(v)$. Then the expected opening cost is

$$E\Big[\sum_{f \in \sigma^*(C') \cup \{r\}} o'(f)\Big] \leq E\Big[\sum_{f \in F^*} o(f)\Big] + M \cdot E\Big[\sum_{v \in C'} c(v, \sigma^*(v))\Big]$$
$$= O^* + M \cdot \frac{\alpha}{M} \cdot \sum_{v \in C} c(v, \sigma^*(v)) = O^* + \alpha\, C^*.$$

The crucial argument here is that we need to account for the extra term $M \cdot c(v, \sigma^*(v))$ only if $v \in C'$, which happens with probability $\frac{\alpha}{M}$.

In order to bound the expected connection cost, we apply Lemma 2, with clients C, core T^*, mapping $\sigma = \sigma^*$, root r, and probability α/M:

$$E\Big[\sum_{v \in C} c(v, \sigma^*(C') \cup \{r\})\Big] \leq \frac{0.807}{\alpha/M} c(T^*) + \sum_{v \in C} c(v, \sigma^*(v)) = \frac{0.807}{\alpha} S^* + C^*.$$

Theorem 2. *Algorithm 1 is an expected* 3.19*-approximation algorithm for CFL.*

Proof. Recall that $\rho_{st} \leq \ln(4) + \varepsilon$ for every fixed $\varepsilon > 0$ [5]. From Lemmas 1, 3, 4, and 5, the total expected cost of the approximate solution is upper bounded by

$$\rho_{st} \cdot (S^* + \alpha \cdot C^*) + \lambda_F(O^* + \alpha\, C^*) + \lambda_C \left(C^* + \frac{0.807}{\alpha} S^*\right)$$
$$= O^* \lambda_F + S^* \left(\rho_{st} + \lambda_C \frac{0.807}{\alpha}\right) + C^*(\rho_{st}\, \alpha + \lambda_F\, \alpha + \lambda_C)$$
$$\overset{\substack{\alpha = 0.539 \\ \lambda_F = 2.294 > 1.67 \\ \lambda_C \leq 1 + 2e^{-\lambda_F}}}{\leq} 2.30 \cdot O^* + 3.19 \cdot S^* + 3.19 \cdot C^* \leq 3.19 \cdot OPT.$$

3 A Constant Approximation for MCFL

Let us consider Algorithm 2 in the figure. With a slight notational abuse, for a set of pairs $\tilde{P}$, we use $\tilde{P}$ also to denote the corresponding set of nodes. In the first step, we define and (approximately) solve a proper PFL instance, whose clients C are given by the nodes in the input pairs P. Currently, $\rho_{pfl} \leq 1.86$ [30]. Let C' be discarded clients, and $\sigma(v)$ be the facility serving $v \in C - C'$. Intuitively, the pairs P_{dir} with at least one endpoint in C' are connected directly via a shortest path. For the remaining pairs $(s, r) \in P_{ind}$, we consider the associated pairs of facilities $(\sigma(s), \sigma(r))$. The latter pairs define an MROB instance `mrob`. To this instance we essentially apply the

Algorithm 2. Approximation algorithm for MCFL

1. Define a PFL instance on the input instance with clients $C = \{s_1, r_1, \ldots, s_k, r_k\}$ (as multi-set), and $p(s) = p(r) = c(s,r)/2$ for any $(s,r) \in P$. Compute a ρ_{pfl}-approximate solution (F', C') for this problem. Let $\sigma(v) \in F'$ be the open facility which is closest to $v \in C - C'$, $P_{dir} := \{(s,r) \in P : \{s,r\} \cap C' \neq \emptyset\}$, and $P_{ind} = P - P_{dir}$.
2. Sample each pair $(\sigma(s), \sigma(r))$, with $(s,r) \in P_{ind}$, independently with probability $1/M$. Let P' be the sampled pairs of facilities.
3. Compute a 2-approximate Steiner forest T' over P' using the primal-dual algorithm in [1].
4. Output (P', T').

MROB algorithm `rand` in [12]. In particular, we sample each pair independently with probability $1/M$, and compute a Steiner forest T' on the sampled pairs P' with the 2-approximation algorithm in [1]. The output solution is given by facilities P' and forest T'.

We need the following result in [12,25]. We recall that, for a set of nodes V' and a forest T', $c_{T'}$ defines distances after contracting the connected components of T', while $c_{V',T'}$ defines distances after contracting the nodes in V' belonging to same tree of T'. In particular, in general $c_{V',T'}(u,v) \geq c_{T'}(u,v)$.

Lemma 6. [12,25] *Consider an MROB instance on pairs $\tilde{P}$, with optimal cost OPT_{mrob}. Sample each pair in $\tilde{P}$ independently with probability $1/M$, and compute a 2-approximate Steiner forest T' on the sampled pairs P' with the algorithm in [1]. Then $E[M \cdot c(T') + \sum_{(s,r)\in\tilde{P}} c_{T'}(s,r)] \leq 5 \cdot OPT_{mrob}$. The same claim holds by replacing $c_{T'}$ with $c_{P',T'}$.*

The last claim of Lemma 6 is not relevant for MROB (it just comes out as a byproduct of the analysis in [12]). However, it is crucial for our analysis. In particular, since P' is a subset of facilities in our case, the connection path for $(\sigma(s), \sigma(r))$ in the MROB solution as given by Lemma 6 enters and leaves trees in T' at facilities. Henceforth, we can extend such connection path with shortest paths $(s, \sigma(s))$ and $(\sigma(r), r)$ to obtain a feasible connection path for pair (s,r).

With a notation analogous to Section 2, we let $OPT = (F^*, T^*)$ denote the optimum solution, where F^* is the set of open facilities and T^* a Steiner forest. We also let $OPT = O^* + C^* + S^*$ be the optimal cost, where O^*, C^*, and S^* are the opening, connection, and Steiner cost, respectively. By O_{pfl}, P_{pfl}, and C_{pfl} we denote, respectively, the opening, penalty, and connection cost of the PFL solution computed in Step 1. We also let $S_{mrob} = M \cdot c(T')$ and $C_{mrob} = \sum_{(s,r)\in P_{ind}} c_{P',T'}(\sigma(s), \sigma(r))$ be the Steiner and connection cost, respectively, of the MROB solution computed in Step 3, as suggested by Lemma 6. Eventually, $APX = O_{apx} + S_{apx} + C_{apx}$ is the cost of the approximate solution, where O_{apx}, S_{apx}, and C_{apx} are the opening, Steiner, and connection cost, respectively.

Lemma 7. $APX \leq O_{pfl} + S_{mrob} + 2P_{pfl} + C_{pfl} + C_{mrob}$.

Proof. By definition, $APX = O_{apx} + S_{apx} + C_{apx}$. Trivially, $O_{apx} \leq O_{pfl}$ (we open a subset P' of the facilities F' in the PFL solution). Moreover, $S_{apx} = S_{mrob}$ by construction.

In order to prove the claim, it is then sufficient to describe connection paths of total cost $2P_{pfl} + C_{pfl} + C_{mrob}$. Let us connect all the pairs in P_{dir} directly via a shortest path. Since by definition at least one endpoint of each pair in P_{dir} belongs to the discarded clients C', $\sum_{(s,r)\in P_{dir}} c(s,r) \leq 2\,P_{pfl}$. Consider now the remaining pairs P_{ind}. For each $(s,r) \in P_{ind}$, we connect s to $\sigma(s)$ and $\sigma(r)$ to r via a shortest path. Then we connect $\sigma(s)$ to $\sigma(r)$ using a shortest path with respect to $c_{P',T'}$. Observe that this is a feasible connection path for (s,r). The total cost of these paths is $\sum_{(s,r)\in P_{ind}} (c(s,\sigma(s)) + c_{P',T'}(\sigma(s),\sigma(r)) + c(\sigma(r),r)) = C_{mrob} + \sum_{v\in P_{ind}} c(v,\sigma(v)) = C_{mrob} + C_{pfl}$. The claim follows.

Lemma 8. $O_{pfl} + C_{pfl} + P_{pfl} \leq \rho_{pfl}(O^* + C^*)$.

Proof. It is sufficient to show that there exists a PFL solution of cost at most $O^* + C^*$. Let $P^*_{dir} \subseteq P$ be the pairs whose connection path in $OPT = (F^*, T^*)$ does not use any edge of T^*, and $P^*_{ind} = P - P^*_{dir}$. By C^*_{dir} (resp., C^*_{ind}) we denote the connection cost of OPT restricted to P^*_{dir} (resp., P^*_{ind}).

Consider the PFL solution (F^*, P^*_{dir}). This solution has opening cost O^* and penalty cost $\sum_{(s,r)\in P^*_{dir}} (p(s)+p(r)) = \sum_{(s,r)\in P^*_{dir}} 2\frac{c(s,r)}{2} = C^*_{dir}$. Moreover, its connection cost is $\sum_{(s,r)\in P^*_{ind}} (c(s,F^*) + c(r,F^*)) \leq \sum_{(s,r)\in P^*_{ind}} c_{F^*,T^*}(s,r) = C^*_{ind}$. Altogether, the cost of this PFL solution is upper bounded by $O^* + C^*_{dir} + C^*_{ind} = O^* + C^*$.

Lemma 9. $E[S_{mrob} + C_{mrob}] \leq 5(C_{pfl} + C^* + S^*)$.

Proof. Let $\tilde{P} := \{(\sigma(s),\sigma(r)) : (s,r) \in P_{ind}\}$ (considered as a multiset). The triple $(\tilde{P}, P', T')$ satisfies the conditions of Lemma 6. Hence, $E[S_{mrob} + C_{mrob}] = E[M \cdot c(T') + \sum_{(s',r')\in\tilde{P}} c_{P',T'}(s',r')] \leq 5\,OPT_{mrob}$, where OPT_{mrob} is the optimum solution to the MROB instance `mrob` induced by pairs $\tilde{P}$.

A bound on OPT_{mrob} is given by the following feasible solution to `mrob`. Buy the edges of the optimal forest T^* of OPT. This costs S^*. For each pair $(\sigma(s),\sigma(r)) \in \tilde{P}$, connect $\sigma(s)$ to s and $\sigma(r)$ to r via a shortest path, and then connect s to r via the connection path between s and r in OPT. The cost of this solution is $\sum_{(s,r)\in P_{ind}}$ $(c(s,\sigma(s))+c(r,\sigma(r))+c_{F^*,T^*}(s,r)) \leq C_{pfl}+C^*$. Then $OPT_{mrob} \leq C_{pfl}+C^*+S^*$. The claim follows.

Theorem 3. *Algorithm 2 is an expected* 16.2*-approximation algorithm for MCFL.*

Proof. One has

$$\begin{aligned}
E[APX] &\overset{\text{Lem 7}}{\leq} O_{pfl} + 2P_{pfl} + C_{pfl} + E[S_{mrob} + C_{mrob}] \\
&\overset{\text{Lem 9}}{\leq} O_{pfl} + 2P_{pfl} + C_{pfl} + 5\,(C_{pfl} + C^* + S^*) \\
&\leq 6\,(O_{pfl} + P_{pfl} + C_{pfl}) + 5\,(C^* + S^*) \\
&\overset{\text{Lem 8}}{\leq} 6\,\rho_{pfl}(O^* + C^*) + 5(C^* + S^*) \\
&\leq (6\,\rho_{pfl} + 5)(O^* + C^* + S^*) \leq 16.2\,OPT.
\end{aligned}$$

4 On the Approximability of SROB

Recall that SROB is the special case of CFL where every node is a facility with opening cost zero. Without loss of generality, we can assume that we are also given a root node $r \in V$ which belongs to the tree T^* in the optimum solution. In this section we show that SROB cannot be approximated within a factor of 1.278, unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log \log n)})$. This heavily improves over the previously known approximation hardness of 1.01 (due to hardness of Steiner tree [6]).

As an intermediate step, we consider a reduction to the *uniform facility location* problem (UnifFL), i.e. the special case of metric facility location where all facilities have uniform opening cost o. Indeed, we consider an even more restrictive case. For a set N of non-negative numbers, let N-UnifFL denote the special case of UnifFL where, for any client $v \in C$ and facility $f \in F$, $c(v, f) \in N$. Given a solution $F' \subseteq F$, we let $\sigma(v)$ denote the facility in F' which is closest to client $v \in C$. We also say that v is *assigned* to $\sigma(v)$ and that $\sigma(v)$ *serves* v.

Guha and Khuller [16] showed that, unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log \log n)})$, $\{1, 3\}$-UnifFL cannot be approximated within a factor of 1.463. While this case seems hard to reduce to SROB, we are able to prove a similar reduction for $\{1, 2\}$-UnifFL.

Lemma 10. *Given an α-approximation algorithm for SROB, there is an α-approximation algorithm for $\{1, 2\}$-UnifFL.*

Proof. Consider a given $\{1, 2\}$-UnifFL instance on clients C and facilities F. First suppose that the uniform opening cost is $o \geq 1$. We define an SROB instance as follows. Consider the complete graph G on nodes $C \cup F \cup \{r\}$, with clients C, root r, and $M = o \geq 1$. Edges (r, f) and (v, f), with $f \in F$, $v \in C$, and $c(v, f) = 1$, have unit costs. All other edges have cost 2. Let the *degree* $d(f) = |\{v \in C : c(v, f) = 1\}|$ of a facility $f \in F$ be the number of clients at distance 1 from that facility. We remark that, if $d(f) \leq M$ for all $f \in F$, then there is an optimum solution of $\{1, 2\}$-UnifFL where only one facility f^* is opened. In fact, suppose $f \neq f^*$ is opened as well, where f serves $x' \leq M$ clients at distance 1 and x'' clients at distance 2. By closing f and assigning its clients to f^*, one saves at least $(o + x' + 2x'') - (2x' + 2x'') = M - x' \geq 0$. The best solution with one open facility can be computed in polynomial time. So we can assume without loss of generality that there is at least one facility f with $d(f) > M$.

Observe that any solution F' to an $\{1, 2\}$-UnifFL instance induces an SROB solution of the same cost. In fact, it is sufficient to consider the tree T' induced by edges $\{r, f\}$, $f \in F'$: this solution costs $|F'| \cdot o + \sum_{v \in C} c(v, F')$. Hence, the cost of an α-approximate solution to SROB costs at most α times the cost of the optimum solution to $\{1, 2\}$-UnifFL.

Thus, it is sufficient to show that any feasible solution to SROB can be turned in polynomial time into a solution to $\{1, 2\}$-UnifFL of not larger cost. Consider any such solution T' to SROB. Suppose that $T' = \emptyset$. Then, adding edge $\{r, f\}$, with f being the maximum degree facility (recall that $d(f) > M$), can only decrease the cost. Next assume $T' \neq \emptyset$. Suppose there is any client v connected either to r or to another client in C. This connection costs 2, thus reconnecting v to any node in $F \cap V(T')$ can only make the solution cheaper. Suppose now that T' contains one edge $\{v, f\}$ with $v \in C$ and $f \in F$, but not edge $\{r, f\}$. Then replacing $\{v, f\}$ by $\{r, f\}$ leaves T' connected

and can only reduce the cost. Finally we may still have edges $\{v, f\}$ and $\{r, f\}$, with $v \in C$ and $f \in F$. Then deleting $\{v, f\}$ again can only decrease the cost, since $M \geq 1$ and no other client (but v) is connected to v.

At the end of the process, T' only contains edges of type $\{r, f\}$, $f \in F$. Let $F' := \{f \in F : \{r, f\} \in T'\}$. F' induces a feasible solution to $\{1, 2\}$-UnifFL of cost equal to the cost of the (modified) SROB solution. The claim follows.

A similar proof holds for the case $o < 1$, by letting $M = 1$ and setting edge costs $\{r, f\}$ to o.

We need the following result by Guha and Khuller [16].

Lemma 11. [16] *Suppose we have a set cover instance* $(\{1, \ldots, n\}, \{S_1, \ldots, S_m\})$ *with unit cost for sets and optimal cost* $OPT_{sc} = k$. *If there is a polynomial time algorithm that can pick* βk *sets (for any constant* $\beta > 0$*) and cover* $c' \cdot n$ *elements, where* $c' > c_\beta = 1 - e^{-\beta}$, *then* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log \log n)})$.

To see why Lemma 11 holds, suppose for the sake of contradiction that such an algorithm does exist. Then we can apply it iteratively to a set cover instance. Let $\alpha_t n$ be the number of covered elements at iteration t, $t = 1, \ldots, T$ (in particular, $\sum_{t=1}^{T} \alpha_t = 1$). At iteration t the algorithm uses only $\beta_t k \leq \delta \cdot \ln(\frac{1}{1-\alpha_t})$ many sets where $\delta < 1$ is a constant. Then we obtain a solution with $\sum_{t=1}^{T} \beta_t k \leq \sum_{t=1}^{T} \delta \ln(\frac{1}{1-\alpha_t})k \leq \delta \cdot \ln(n)k$ sets, contradicting the hardness result of [10].

Lemma 12. *There is no* α*-approximation algorithm with* $\alpha \leq 1.278$ *for* $\{1, 2\}$*-UnifFL, unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log \log n)})$.

Proof. Suppose for the sake of contradiction that we have an α-approximation algorithm for $\{1, 2\}$-UnifFL and $\alpha \leq 1.278$. Consider a set cover instance with n elements, sets $S_1, \ldots, S_m$ of unit cost, and optimal value $OPT_{sc} = k$. Define a $\{1, 2\}$-UnifFL instance as follows. Introduce a facility f for each set S_f and a client v for each element v. Set $c(v, f) = 1$ if $v \in S_f$, and $c(v, f) = 2$ otherwise. Let $o = \gamma \frac{n}{k}$ be the uniform cost. Here $0 < \gamma < 1$ is a constant, that we will determine later. Let OPT_{ufl} be the optimum solution to this instance. By opening the k facilities which correspond to the k sets in the optimum set cover solution, we obtain that $OPT_{ufl} \leq k \cdot o + 1 \cdot n = k \cdot \gamma \frac{n}{k} + n = (1 + \gamma)n$.

Using the α-approximation algorithm we get a $\{1, 2\}$-UnifFL solution F' of cost at most $\alpha(1 + \gamma)n$. Let $c \in [0, 1]$ be the fraction of demands, whose service costs are 1 (the others are served at cost 2). Define $\beta := |F'|/k > 0$. Then

$$(\beta\gamma + 2 - c)n = \underbrace{\beta k}_{=|F'|} \cdot o + \underbrace{cn + 2 \cdot (1 - c)n}_{\text{connection cost}} \leq \alpha(1 + \gamma)n.$$

This can be rearranged to

$$c \geq \beta \cdot \gamma + 2 - \alpha(1 + \gamma) \overset{\alpha \leq 1.278, \gamma := 0.278}{\geq} 0.278\beta + 0.3667 > 1 - e^{-\beta},$$

contradicting Lemma 11.

Theorem 4. *There cannot be a factor* 1.278*-approximation algorithm for SROB, MROB, CFL, MCFL, SSBB, and VPN, unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log \log n)})$.

Proof. The claim for SROB follows from Lemmas 10 and 12. The same result trivially extends to MROB, CFL, MCFL, and SSBB (which are all generalizations of SROB). The hardness for VPN follows from the fact, that any SROB instance is equivalent to a VPN instance with a single receiver of capacity M (see, e.g., [15,27]).

Acknowledgments. A special thank to J. Byrka, S. Leonardi, and M. Singh for helpful discussions.

References

1. Agrawal, A., Klein, P., Ravi, R.: When trees collide: an approximation algorithm for the generalized Steiner problem on networks. SIAM Journal on Computing 24, 440–456 (1995)
2. Awerbuch, B., Azar, Y.: Buy-at-bulk network design. In: FOCS, pp. 542–547 (1997)
3. Becchetti, L., Könemann, J., Leonardi, S., Pál, M.: Sharing the cost more efficiently: improved approximation for multicommodity rent-or-buy. In: SODA, pp. 375–384 (2005)
4. Byrka, J.: An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) RANDOM 2007 and APPROX 2007. LNCS, vol. 4627, pp. 29–43. Springer, Heidelberg (2007)
5. Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L.: An improved LP-based approximation for Steiner tree. In: STOC, pp. 583–592 (2010)
6. Chlebík, M., Chlebíková, J.: The Steiner tree problem on graphs: Inapproximability results. Theoretical Computer Science 406(3), 207–214 (2008)
7. Eisenbrand, F., Grandoni, F.: An improved approximation algorithm for virtual private network design. In: SODA, pp. 928–932 (2005)
8. Eisenbrand, F., Grandoni, F., Oriolo, G., Skutella, M.: New approaches for virtual private network design. SIAM Journal on Computing 37(3), 706–721 (2007)
9. Eisenbrand, F., Grandoni, F., Rothvoß, T., Schäfer, G.: Connected facility location via random facility sampling and core detouring. Journal of Computer and System Sciences 76, 709–726 (2010)
10. Feige, U.: A Threshold of ln n for Approximating Set Cover. Journal of the ACM 45(4) (1998)
11. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. Journal of Computer and System Sciences 69(3), 485–497 (2004)
12. Fleischer, L., Könemann, J., Leonardi, S., Schäfer, G.: Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In: STOC, pp. 663–670 (2006)
13. Garg, N., Khandekar, R., Konjevod, G., Ravi, R., Salman, F., Sinha, A.: On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design problem. In: Aardal, K., Gerards, B. (eds.) IPCO 2001. LNCS, vol. 2081, pp. 170–184. Springer, Heidelberg (2001)
14. Grandoni, F., Italiano, G.F.: Improved approximation for single-sink buy-at-bulk. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288, pp. 111–120. Springer, Heidelberg (2006)
15. Grandoni, F., Rothvoß, T.: Network design via core detouring for problems without a core. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 490–502. Springer, Heidelberg (2010)
16. Guha, S., Khuller, S.: Greedy Strikes Back: Improved Facility Location Algorithms. Journal of Algorithms 31(1), 228–248 (1999)

17. Guha, S., Meyerson, A., Munagala, K.: A constant factor approximation for the single sink edge installation problem. SIAM Journal on Computing 38(6), 2426–2442 (2009)
18. Gupta, A., Kleinberg, J., Kumar, A., Rastogi, R., Yener, B.: Provisioning a virtual private network: a network design problem for multicommodity flow. In: STOC, pp. 389–398 (2001)
19. Gupta, A., Kumar, A.: A constant-factor approximation for stochastic Steiner forest. In: STOC, pp. 659–668 (2009)
20. Gupta, A., Kumar, A., Pal, M., Roughgarden, T.: Approximation via cost-sharing: simpler and better approximation algorithms for network design. Journal of the ACM 54(3), 11 (2007)
21. Gupta, A., Srinivasan, A., Tardos, É.: Cost-sharing mechanisms for network design. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) RANDOM 2004 and APPROX 2004. LNCS, vol. 3122, pp. 139–150. Springer, Heidelberg (2004)
22. Jothi, R., Raghavachari, B.: Improved approximation algorithms for the single-sink buy-at-bulk network design problems. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, pp. 336–348. Springer, Heidelberg (2004)
23. Karger, D.R., Minkoff, M.: Building Steiner trees with incomplete global knowledge. In: FOCS, pp. 613–623 (2000)
24. Kumar, A., Gupta, A., Roughgarden, T.: A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. In: FOCS, pp. 333–342 (2002)
25. Leonardi, S.: Private communication (2008)
26. Meyerson, A., Munagala, K., Plotkin, S.: Cost-distance: two metric network design. In: FOCS, pp. 624–630 (2000)
27. Rothvoß, T., Sanità, L.: On the complexity of the asymmetric VPN problem. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX 2009. LNCS, vol. 5687, pp. 326–338. Springer, Heidelberg (2009)
28. Swamy, C., Kumar, A.: Primal–dual algorithms for connected facility location problems. Algorithmica 40(4), 245–269 (2004)
29. Talwar, K.: The single-sink buy-at-bulk LP has constant integrality gap. In: Cook, W.J., Schulz, A.S. (eds.) IPCO 2002. LNCS, vol. 2337, pp. 475–480. Springer, Heidelberg (2002)
30. Xu, G., Xu, J.: An improved approximation algorithm for uncapacitated facility location problem with penalties. Journal of Combinatorial Optimization 17(4), 424–436 (2009)

Safe Lower Bounds for Graph Coloring

Stephan Held*, William Cook**, and Edward C. Sewell

University of Bonn, Georgia Institute of Technology,
Southern Illinois University Edwardsville
held@or.uni-bonn.de, bico@isye.gatech.edu,
esewell@siue.edu

Abstract. The best known method for determining lower bounds on the vertex coloring number of a graph is the linear-programming column-generation technique first employed by Mehrotra and Trick in 1996. We present an implementation of the method that provides numerically safe results, independent of the floating-point accuracy of linear-programming software. Our work includes an improved branch-and-bound algorithm for maximum-weight stable sets and a parallel branch-and-price framework for graph coloring. Computational results are presented on a collection of standard test instances, including the unsolved challenge problems created by David S. Johnson in 1989.

Keywords: graph coloring, fractional chromatic number, column generation, maximum-weight stable set, safe computations.

1 Introduction

Let $G = (V, E)$ be an undirected graph with a set V of vertices and a set E of edges. We follow the usual notation $n = |V|$ and $m = |E|$. A *stable set* is a subset $S \subset V$ composed of pairwise non-adjacent vertices, that is, $\{v, w\} \notin E$ for all $v, w \in S$. A *coloring* of G, or a *k-coloring*, is a partition of V into k stable sets $S_1, \ldots, S_k$. The minimum k such that a k-coloring exists in G is called the *chromatic number* of G and is denoted $\chi(G)$.

A *clique* is a subset $C \subset V$ composed of pairwise adjacent vertices, that is, $\{v, w\} \in E$ for all $v, w \in C$. The *clique number* $\omega(G)$, defined as the size of a largest clique in G, is a lower bound for $\chi(G)$. Similarly, the *stability number* $\alpha(G)$, defined as the maximum size of a stable set in G, provides another lower bound $\lceil n/\alpha(G) \rceil \leq \chi(G)$.

Letting $\mathcal{S}$ denote the set of all maximal stables sets in G, it is well known that $\chi(G)$ is the optimal value of following integer-programming problem (e.g.

* Research supported by a postdoctoral fellowship grant from the DAAD.
** Research supported by NSF Grant CMMI-0726370 and ONR Grant N00014-08-1-1104.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 261–273, 2011.

see [13])

$$\begin{aligned} \chi(G) = \min \quad & \sum_{S \in \mathcal{S}} x_S & \\ s.t. \quad & \sum_{S \in \mathcal{S}: v \in S} x_S \geq 1 & \forall v \in V \\ & x_S \in \{0,1\} & \forall S \in \mathcal{S}. \end{aligned} \tag{CIP}$$

The optimal value, $\chi_f(G)$, of the linear-programming (LP) relaxation, denoted by (CLP), obtained by replacing the integrality condition by $0 \leq x_S \leq 1$ for all $S \in \mathcal{S}$, is called the *fractional chromatic number* of G. It defines the lower bound $\lceil \chi_f(G) \rceil$ for $\chi(G)$. In [10] it was shown that the integrality gap between $\chi(G)$ and $\chi_f(G)$ is $\mathcal{O}(\log\ n)$ wherefore it is *NP*-hard to compute $\chi_f(G)$. In fact, for $\chi(G)$ as well as for $\chi_f(G)$ and all $\epsilon > 0$ there does not exist a polynomial-time approximation algorithm that achieves an approximation ratio of n^ϵ unless $P = NP$ [22].

Mehrotra and Trick [13] proposed to solve (CLP) via *column generation* and, accordingly, (CIP) via *branch and price*. Their process is the most successful exact coloring method proposed to date, including impressive results obtained recently by Gualandi and Malucelli [8] and by Malaguti, Monaci, and Toth [11].

Our study focuses on an implementation of Mehrotra-Trick that is guaranteed to produce correct results, independent of the floating-point accuracy of LP software employed in the computation.

To see the possible difficulty, consider the `queen16_16` instance from the DIMACS test collection. Using the state-of-the art LP solver Gurobi 3.0.0, at the termination of the column-generation process the floating-point representation $\chi_f^{float}(G)$ of the fractional chromatic number for this graph is $\chi_f^{float}(G) = 16.0000000000001315$. But $17 = \lceil 16.0000000000001315 \rceil$ is not a valid lower bound, since there is a known 16-coloring for `queen16_16`. In general, $\chi_f^{float}(G)$ can be smaller than, equal to, or larger than $\chi_f(G)$. This difficulty is compounded by the need to accurately run the column-generation process when dual LP solutions are available only as floating-point approximations.

We propose a technique to avoid this inaccuracy by computing a numerically safe lower bound on $\chi_f(G)$, using a floating-point LP solution as a guide. To drive the process, we also present a new combinatorial branch-and-bound algorithm to compute maximum-weight stable sets in graphs; the new method is particularly well suited for the instances of the problem that arise in the Mehrotra-Trick procedure. With this safe methodology, we are able to verify results reported in previous studies as well to obtain new best known bounds for a number of instances from the standard DIMACS test collection. In particular, we have improved previously reported results on four of the eight open `DSJCxxx` instances created by David S. Johnson in 1989, including the optimal solution of `DSJC250.9`.

2 Column Generation

Let $\mathcal{S}' \subseteq \mathcal{S}$ contain a feasible solution to (CLP), that is, $V = \bigcup_{S \in \mathcal{S}'} S$, and consider the restricted LP problem defined as

$$\begin{aligned} \chi_f(G, \mathcal{S}') := \min \quad & \sum_{S \in \mathcal{S}'} x_S \\ s.t. \quad & \sum_{S \in \mathcal{S}' : v \in S} x_S \geq 1 \quad \forall v \in V \\ & 0 \leq x_S \leq 1 \; \forall S \in \mathcal{S}'. \end{aligned} \tag{CLP-r}$$

Let (x, π) be an optimum primal-dual solution pair to (CLP-r), where the dual solution vector $\pi = (\pi_v)_{v \in V}$ contains a value $\pi_v \in [0, 1]$ for every $v \in V$. By setting $x_S = 0$ for all $S \in \mathcal{S} \setminus \mathcal{S}'$, x can be extended naturally to a feasible solution of (CLP). Now, either (x, π) is also optimum or π is dual infeasible with respect to (CLP). In the latter case, there is a stable set $S \in \mathcal{S} \setminus \mathcal{S}'$ with

$$\pi(S) > 1, \tag{1}$$

where we use the notation $\pi(X) := \sum_{v \in X} \pi_v$ for a subset $X \subseteq V$. A stable set satisfying (1) exists if and only if the *weighted stability number*

$$\begin{aligned} \alpha_\pi(G) := \max \quad & \sum_{v \in V} \pi_v y_v \\ s.t. \quad & y_v + y_w \leq 1 \qquad \forall \{v, w\} \in E \\ & y_v \in \{0, 1\} \qquad \forall V \in V \end{aligned} \tag{MWSS}$$

is greater than one.

2.1 Finding Maximum-Weight Stable Sets

The maximum-cardinality stable-set problem and its weighted version (MWSS) are among the hardest combinatorial optimization problems. For any $\epsilon > 0$, $\alpha(G)$ cannot be approximated within a factor $O(n^{1-\epsilon})$ unless $P{=}\,NP$ [22]. However, for very dense graphs, for example with edge-density $\rho(G) := m/(n(n-1)/2) \sim 0.9$, the size and number of maximal stable sets is quite low and can be enumerated. A particularly efficient way of solving (MWSS) in dense graphs is via Östergård's CLIQUER algorithm [23], which we employ on dense instances. For sparse graphs CLIQUER becomes less efficient and for such instances we employ a new algorithm described below.

Combinatorial Branch and Bound. The branch-and-bound algorithm presented here uses depth-first search and adopts ideas from algorithms presented in [4,3,18,21].

A subproblem in the branch-and-bound tree consists of a lower bound, denoted LB, which is the weight of the heaviest stable set found so far, the current stable set $S = \{v_1, v_2, \ldots, v_d\}$ (where d is the depth of the subproblem in the search tree), the set of free vertices F, and a set of vertices X that are excluded from the current subproblem (which will be explained below). The goal of the subproblem is to either prove that this subproblem cannot produce a heavier stable set than the heaviest one found so far (that is, $\pi(S) + \alpha_\pi(G[F]) \leq LB$) or find a maximum-weight stable set in $G[F]$ (given a vertex set $W \subseteq V$, its induced subgraph $G[W]$ is defined as $G[W] := (W, \{\{v, w\} \in E : v, w \in W\})$).

An overview is given in Algorithm 1. The algorithm consists of a recursive subfunction MWSS_RECURSION(S, F, X) that is called with $S = \emptyset, F = V$ and $X = \emptyset$.

Algorithm 1. An Exact Maximum-Weight Stable Set Algorithm

```
function MWSS_RECURSION(S,F,X)
    LB = max(LB, π(S));
    if F = ∅ then return;
    end if
    if ∃ x ∈ X with π_x ≥ π((S ∪ F) ∩ N(x)) then return;
    end if
    Find a weighted clique cover of G[F];
    if weight of the clique cover ≤ LB − π(S) then return;
    end if
    Determine the branch vertices F'' = {f_1, f_2, ..., f_p} ⊂ F
        using the three branching rules;
    for i = p down to 1 do
        F_i = F \ (N(f_i) ∪ {f_i, f_{i+1}, ..., f_p});
        MWSS_RECURSION(S ∪ {f_i}, F_i, X);
        X = X ∪ {f_i};
    end for
end function
MWSS_RECURSION(∅, V, ∅);
```

The algorithm uses two methods to prune subproblems. The first method works as follows. Let X be the set of vertices that have been excluded from consideration in the current subproblem because they have already been explored in an ancestor of the current subproblem (see Algorithm 1 to see how X is created). If there exists a vertex $x \in X$ such that $\pi_x \geq \pi((S \cup F) \cap N(x))$, then the current subproblem cannot lead to a heavier stable set than has already been found. To see this, let S' be the heaviest stable set that can be created by adding vertices from F to S. Now consider the stable set $S'' = \{x\} \cup S' \backslash N(x)$ created by adding x to S' and removing any of its neighbors from S'. Then

$$\begin{aligned} \pi(S'') &= \pi(\{x\} \cup S' \backslash N(x)) && = \pi_x + \pi(S' \backslash N(x)) \\ &= \pi_x + \pi(S') - \pi(S' \cap N(x)) && \geq \pi_x + \pi(S') - \pi((S \cup F) \cap N(x)) \\ &\geq \pi(S'), \end{aligned}$$

where the second to last inequality follows from the fact that S' is contained in $S \cup F$ and the last inequality follows from the supposition that

$$\pi_x \geq \pi((S \cup F) \cap N(x)).$$

Furthermore, every vertex in S'' was available when x was explored as a branch vertex, thus LB must have been greater than or equal to $\pi(S'')$ when the algorithm returned from exploring x as the branch vertex. Consequently, $LB \geq \pi(S'') \geq \pi(S')$. Hence, this subproblem can be pruned.

The second method of pruning subproblems uses weighted clique covers. A *weighted clique cover* for a set of vertices F is a set of cliques $K_1, K_2, \ldots, K_r$ together with a positive weight Π_i for each clique K_i such that $\sum_{i:f\in K_i} \Pi_i \geq \pi_f$ for each vertex $f \in F$. The *weight of the clique cover* is defined to be $\sum_{i=1}^{r} \Pi_i$. It is easy to show that $\alpha_\pi(G[F])$ is less than or equal to the weight of any clique cover of F. Hence, if a clique cover of weight less than or equal to $LB - \pi(S)$ can be found for F, then this subproblem can be pruned.

An iterative heuristic is used to find weighted clique covers. The heuristic repeatedly chooses the vertex v with the smallest positive weight, finds a maximal clique K_i that contains v, assigns the weight $\Pi_i = \pi_v$ to K_i, and subtracts Π_i from the weight of every vertex in K_i.

The algorithm uses three branching rules to create subproblems. The first two rules adopt a weighted variation of a technique employed by Balas and Yu [5,4]. Suppose that $F' \subseteq F$ and it can be proved that

$$\alpha_\pi(G[F']) \leq LB - \pi(S).$$

Let $F'' = F \backslash F' = \{f_1, f_2, \ldots, f_p\}$ and let $F_i = F \backslash (N(f_i) \cup \{f_i, f_{i+1}, \ldots, f_p\})$. If $\alpha_\pi(G[F]) > LB - \pi(S)$, then

$$\alpha_\pi(G[F]) = \max_{i=1,\ldots,p} \pi_{f_i} + \alpha_\pi(G[F_i]).$$

Hence, one branch is created for each set $F_1, \ldots, F_p$.

The first branching rule uses the weighted clique cover to create F'. The clique cover heuristic is halted as soon as the weight of the clique cover would exceed $LB - \pi(S)$. Then F' is defined as the set of vertices whose remaining weight is zero (that is, $F' = \left\{f \in F : \pi'_f = 0\right\}$) and $F'' = F \backslash F'$.

The second branching rule uses a method similar to the first method of pruning. If there exists a vertex $x \in X$ such that $\pi_x \geq \pi(S \cap N(x))$, then it can be shown that if $\alpha_\pi(G[F]) > LB - \pi(S)$, then every maximum-weight stable set in $G[F]$ must contain at least one neighbor of x that is in F. The proof is similar to the proof for the first method of pruning. In such a case, F'' is set equal to $N(x) \cap F$.

The third branching rule searches for a vertex $f \in F$ such that $\pi_f \geq \pi(F \cap N(f))$. If such a vertex exists, it is easy to prove that there exists an maximum-weight stable set of $G[F]$ that includes f, hence a single branch is created (that is, $F'' = \{f\}$).

The algorithm uses the rule that generates the smallest F'' (ties are broken in favor of the first rule and then the third rule). For both the second and the third branching rules, the set of vertices F'' are sorted in increasing order of their degree in $G[F]$.

In the context of column generation the running time can be reduced further because the actual maximum-weight stable set need not necessarily be found. Instead, it is sufficient to either find a stable set S with $\pi(S) > 1$ or decide that no such set exists.

Hence, LB can be initialized as 1, because only solutions of value bigger than one are of interest. Furthermore, it is sufficient to stop the algorithm once a stable set S with $\pi(S) > 1$ is found.

Heuristics. Within the column-generation process, a stable set with $\pi(S) > 1$ can often be found by heuristic methods. The heuristics we use create an initial solution by a greedy strategy and then improve this solution with local search. The greedy algorithms build a stable set $S \in \mathcal{S} \setminus \mathcal{S}'$ by starting with an empty set and adding vertices one by one. A vertex $v \in V \setminus S$ is added to S if $S \cup \{v\}$ is a stable set. Mehrotra and Trick proposed to traverse the vertices in non-decreasing order of their weight [13]. We use the following three greedy orderings: as the next vertex, try a not yet processed vertex $v \in V \setminus (N(S) \cup S)$ for which

1. π_v (maximum weight strategy)
2. $\pi_v - \sum_{w \in N(v) \setminus N(S)} \pi_w$ (maximum dynamic surplus strategy)
3. $\pi_v - \sum_{w \in N(v)} \pi_w$ (maximum static surplus strategy)

is maximum.

The result of the greedy algorithm is then improved by local search similar to the local swaps in [1]. If we do not find a stable set of weight greater than one, then we perform several additional searches using slightly perturbed greedy orders.

3 Numerically Safe Bounds

Competitive LP codes for solving (CLP-r) use floating-point representations for all numbers. This causes immediate difficulties in the column-generation process. Indeed, let π^{float} denote the vector of dual variables in floating-point representation as returned by an LP-solver. Based on these inexact values, $\alpha_\pi(G) > 1$ can hardly be decided and this can lead to premature termination or to endless loops (if the same stable set is found again and again).

One way to circumvent these problems would be to solve (CLP-r) exactly, for example with a solver such as [2]. However, exact LP-solvers suffer significantly higher running times, and in column generation, where thousands of restricted problems must be solved, these solvers would be impractical. Thus, instead of computing $\chi_f(G)$ exactly, we compute a numerically safe lower bound $\underline{\chi_f}(G)$ in exact integer (fixed point) arithmetic, where the floating-point variables π^{float} serve only as a guide.

Recall that any vector $\pi \in [0,1]^n$, with $\alpha_\pi(G) \leq 1$ is a dual feasible solution of (CLP) and defines a lower bound regardless whether it is optimum or not. Accordingly, given a scale factor $K > 0$, a vector $\pi^{int} \in \mathbb{N}_0^{V(G)}$ proves the lower bound $K^{-1}\pi^{int}(V)$ if and only $\alpha_{\pi^{int}}(G) \leq K$.

Now, the goal is to conduct the maximum-weight stable-set computations with integers $\pi_v^{int} := \lfloor K\pi_v^{float} \rfloor$ $(v \in V)$. Thus, achieving a lower $\frac{n}{K}$-approximation of $\pi_v^{float}(V)$:

$$\pi_v^{float}(V) - \frac{n}{K} \leq \frac{1}{K}\pi_v^{int}(V) \leq \pi_v^{float}(V). \tag{2}$$

The question is how to represent the integers π_v^{int} $(v \in V)$ and how to choose K. For performance reasons, it is preferable to use integer types that are natively supported by the computer hardware, e.g. 32- or 64-bit integers in two's complement.

More generally, let us assume that all integers are restricted to an interval $[I_{\min}, I_{\max}]$ with $I_{\min} < 0$ and $I_{\max} > 0$. To avoid integer overflows, we have to ensure that during the computations of maximum-weight stable sets the intermediate results neither fall below $I_{\min}$ nor exceed $I_{\max}$. The smallest intermediate results occur while computing surpluses with the greedy strategies 2 and 3. The largest intermediate results are either given by $\pi^{int}(X)$ for some $X \subset V$ or as the weight of the weighted clique covers in Algorithm 1. As $\pi_v^{float} \in [0,1]$ $(v \in V)$, setting $K := \min\{-I_{\min}, I_{\max}\}/n$ guarantees that any intermediate result will be representable within $[I_{\min}, I_{\max}]$. Note that the dual variables returned as floating point numbers by the LP solver might exceed the permissible interval $[0,1]$ slightly. They are shifted into $[0,1]$ before scaling.

By (2) the deviation from the floating-point representation of the fractional chromatic number is at most $n^2/\min\{-I_{\min}, I_{\max}\}$. Note that the denominator grows exponentially in the number of bits that are spent to store numbers, allowing a reduction in the error without much memory overhead.

Column generation creating a safe lower bound is summarized in Algorithm 2. Initially, a coloring is determined with the greedy algorithm DSATUR [6]. It provides the initial set $\mathcal{S}'$ and an upper bound for $\chi(G)$.

The column-generation process terminates when $\alpha_{\pi^{int}}(G) \leq K$ with a lower bound of $\underline{\chi_f}(G) := K^{-1}\pi^{int}(V)$
$\leq \chi_f(G)$.

Note that it is difficult to bound the difference $\chi_f(G) - \underline{\chi_f}(G)$ without further assumptions on the LP solver. However, a close upper bound $\overline{\chi_f}(G)$ for $\chi_f(G)$ can

Algorithm 2. Column Generation for Computing $\underline{\chi_f}(G)$

$\mathcal{S}' \leftarrow$ Compute initial coloring (DSATUR).
$S \leftarrow \emptyset$
repeat
  $\mathcal{S}' \leftarrow \mathcal{S}' \cup S$
  $\pi^{float} \leftarrow$ Solve (CLP-r) in floating-point arithmetic
  $\pi^{int} \leftarrow \lfloor K\pi^{float} \rfloor$
  $S \leftarrow$ search for an improving stable set by heuristic (Section 2.1) or Algorithm 1
until $\pi^{int}(S) \leq K$
$\underline{\chi_f}(G) \leftarrow K^{-1}\pi^{int}(V)$

be computed by solving the final restricted LP (CLP-r) once in exact arithmetic [2]. Thereby, an interval $[\underline{\chi_f}(G), \overline{\chi_f}(G)]$ containing $\chi_f(G)$ can be determined, allowing us to obtain the precise value of $\lceil \chi_f(G) \rceil$ on most test instances.

4 Improved Computation of Lower Bounds

4.1 Decreasing Dual Weights for Speed

If the weight of a maximum-weight stable set in Algorithm 2 is slightly larger than K, it can potentially be reduced to K, or less, by decreasing the integer variables π^{int}. This way an earlier termination of the column-generation approach might be possible. Of course, such reduced weights will impose a lower fractional bound. However, the entries of π^{int} can be reduced safely by a total amount of

$$\operatorname{frac}(\pi^{int}, K) := \max\left\{0, \left(\sum_{v \in V} \pi_v^{int} - 1\right)\right\} \mod K, \tag{3}$$

while generating the same lower bound of $\lceil K^{-1}\pi^{int}(V) \rceil$. The difficulty is to decide how to decrease entries in π_v^{int}. Ideally, one would like to achieve a largest possible ratio between the reduction of the value of the maximum-weight stable set and the induced lower bound for the chromatic number.

Gualandi and Malucelli [8] proposed a *uniform rounding* style, rounding down all values π_v^{int} $(v \in V)$ uniformly by $\operatorname{frac}\left(\pi^{int}, K\right)/n$. This way the weight of a stable set $S \in \mathcal{S}$ decreases by $\frac{|S|}{n} \operatorname{frac}(\pi^{int}, K)$.

An alternative technique works as follows. Consider a $v \in V$ with $\pi_v > 0$, then at least one vertex from $V' := v \cup \{w \in N(v) \,:\, \pi_w > 0\}$ will be contained in a maximum-weight stable set. Thus, to reduce the value of the maximum-weight stable set, it is sufficient to reduce weights in V' only. In our implementation, we always select a set V' of smallest cardinality. We refer to this rounding style as *neighborhood rounding*.

Table 1. Impact of reducing dual weights on # calls to Algorithm 1

Instance	$\lvert V\rvert$	$\lvert E\rvert$	None	Uniform	Neighborhood
latin_square_10	900	307350	1	1	1
queen16_16	256	12640	1	1	1
1-Insertions_5	202	1227	67	1	1
1-Insertions_6	607	6337	> 18046	9	40
DSJC250.1	250	3218	> 301	1	1
DSJC250.5	250	15668	18	13	13
DSJC500.5	500	62624	75	39	38
flat300_28_0	300	21695	25	5	4
myciel7	191	2360	79	33	5

Table 1 demonstrates the importance of rounding for instances from the DIMACS benchmark set, covering several instance classes. It reports the number of calls of the exact maximum-weight stable-set solver (Algorithm 1) needed to terminate column generation, in column 4 without any dual weight reduction (beyond safe weights according to Section 3), in column 5 with uniform rounding, and in column 6 with neighborhood rounding. However, neither of the two dual variable reduction styles dominates the other.

5 Experimental Results

The described algorithms were implemented in the C programming language; our source code is available online [9]. The LP problems (CLP-r) are solved with Gurobi 3.0.0 in double floating-point precision. Experiments were carried out on the DIMACS graph-coloring instances [20], using a 2.268 GHz Intel Xeon E5520 server, compling with gcc -O3. To compute $\overline{\chi_f}(G)$ by solving (CLP-r) exactly we used the exact LP-solver QSopt_ex [2].

5.1 Results of Column Generation

We were able to compute $\underline{\chi_f}(G)$ and $\overline{\chi_f}(G)$ for 119 out of 136 instances, limiting the running time for computing $\underline{\chi_f}(G)$ to three days per instance. Solving (CLP-r) exactly can be quite time consuming, for example, on `wap02a` it takes 34 hours, compared to 10 minutes in doubles (using QSopt_ex in both cases). This demonstrates that the use of an exact LP-solver for every instance of (CLP-r) would be impractical. As we compute $\overline{\chi_f}(G)$ only for the academic purpose of estimating the differences $\chi_f(G) - \underline{\chi_f}(G)$, we do not report its running times from here on.

For all the 119 solved DIMACS instances it turned out that $\lceil \underline{\chi_f}(G) \rceil = \lceil \overline{\chi_f}(G) \rceil$. Thus, we obtained safe results for $\lceil \chi_f(G) \rceil$. But there were many instances where $\overline{\chi_f}(G) < \pi^{float}(V)$, and the floating-point solutions implied by the LP-solver would have been wrong, as in the example from Section 3: queen16_16. However, we did not find previously reported results for $\lceil \chi_f(G) \rceil$ that were incorrect.

Here, we focus on those instances for which the chromatic number is or was unknown. For space reasons, we skip those open `*-Insertions_*` and `*-FullIns_*` instances where $\lceil \chi_f(G) \rceil$ was already reported in [8] or [11]. Table 2 shows the results on the remaining open instances. Columns 2 and 3 give the number of vertices and edges, column 4 shows $\lceil \chi_f(G) \rceil$ from our computations, where bold numbers are those where we could improve best-known lower bounds. Column 5 shows the clique numbers from the literature or computed with CLIQUER, columns 6 and 7 summarize the best lower and upper bounds that can be found in the literature [7,8,15,16,17,19]. The last column shows the running time for computing $\underline{\chi_f}(G)$.

For the instances `DSJC500.5`, `DSJC1000.5`, `flat1000_50_0`, `flat1000_60_0`, `flat1000_76_0`, `wap01a`, `wap02a`, `wap07a`, `wap08a` , and `3-Insertions_5` we

Table 2. Computational results on open benchmarks

Instance	$\|V\|$	$\|E\|$	$\lceil\chi_f(G)\rceil$	$\omega(G)$	old LB	old UB	Time (sec.)
DSJC250.5	250	15668	26	12	26[8]	28[8]	18
DSJC250.9	250	27897	71	42	71[8]	72[8]	8
DSJC500.1	500	12458	*	5	6[7]	12[17]	*
DSJC500.5	500	62624	**43**	13	16[7]	48[17]	439
DSJC500.9	500	224874	123	54	123[8]	126[17]	100
DSJC1000.1	1000	49629	*	6	6[7]	20[17]	*
DSJC1000.5	1000	249826	**73**	14	17[7]	83[17]	142014
DSJC1000.9	1000	449449	215	63	215[8]	222[19]	5033
r1000.1c	1000	485090	96	89	96[8]	98[8]	2634
C2000.5	2000	999836	*	16	16	148[17]	*
C4000.5	4000	4000268	*	≥ 17	17	271[17]	*
latin_square_10	900	307350	90	90	90[15]	98[12]	76
abb313GPIA	1557	65390	8	8	8[15]	9[11]	3391
flat1000_50_0	1000	245000	**50**	14	14	50[8]	3331
flat1000_60_0	1000	245830	**60**	14	14	60[8]	29996
flat1000_76_0	1000	246708	**72**	14	14	82[8]	190608
wap01a	2368	110871	41	41	41[15]	43[11]	20643
wap02a	2464	111742	40	40	40[15]	42[11]	236408
wap03a	4730	286722	*	40	40[15]	47[11]	*
wap04a	5231	294902	*	40	40[15]	44[11]	*
wap07a	1809	103368	40	40	40[15]	42[11]	25911
wap08a	1870	104176	40	40	40[15]	42[11]	18015
3-Insertions_5	1406	9695	3	2	3[15]	6[15]	6959

could compute $\lceil\chi_f(G)\rceil$ for the first time, improving known lower bounds on DSJC500.5, DSJC1000.5, flat1000_50_0, flat1000_60_0, and flat1000_76_0 significantly. On flat1000_50_0 and flat1000_60_0, $\lceil\chi_f(G)\rceil$ proves the optimality of known upper bounds.

On most instances that are not listed $\chi_f(G)$ is computed much faster than within three days. The geometric mean of the running times of the 119 solved instances is 6.5 seconds. 17 DIMACS instances were not finished within three days. For 11 of these instances (le450_5a, le450_5b, le450_5c, le450_5d, le450_15a, le450_15b, le450_15c, le450_15d, le450_25c, and le450_25d, and qg.order100) the clique numbers $\omega(G)$ can be computed within seconds by CLIQUER [23] and match known upper bounds and proving $\omega(G) = \chi_f(G) = \chi(G)$.

5.2 Results of Branch and Price

For all open benchmark instances, we attempted to improve the lower bounds by branch and price as described in [13], allowing a time limit of three days. This way we could improve the lower bounds of DSJC1000.9 and DSJC250.9 by one to 216 and **72** respectively, proving optimality of a known upper bound for DSJC250.9.

We also did excessive branching experiments with up to 60 parallel processors, but for other instances the lower bounds grow too slow to achieve better integral bounds within a few weeks.

5.3 Results on Dense Subgraphs

As already noted in Section 5.1, for 17 very large DIMACS instances we were not able to compute $\lceil \chi_f(G) \rceil$. For 11 of these instances, $\omega(G)$ is easy to compute and yields a tight lower bound. For each of the remaining six instances `DSJC500.1`, `DSJC1000.1`, `C2000.5`, `C4000.5`, `wap03a`, and `wap04a` the gap between the published lower and upper bounds is particularly large.

However, on these instances column generation can still be applied if restricted to tractable subgraphs. It is easy to see that for any subgraph $G[X]$ induced by $X \subset V$ (see Section 2.1), $\chi_f(G[X]) \leq \chi_f(G)$ and, thus, $\lceil \chi_f(G[X]) \rceil$ imposes a lower bound for $\chi(G)$. The set X should be chosen such that $\lceil \chi_f(G[X]) \rceil$ is large, but still solvable. For the first goal a dense subgraph $G[X]$ would be favorable. We use a simple greedy strategy that starts with $X = V$ and iteratively deletes a vertex of minimum degree until $|X|$ has a given size.

Table 3 shows the lower bounds, we could obtain this way. Columns 2 and 3 give the sizes of the induced subgraph. Column 4 reports the lower bounds obtained from the subgraphs, while column 5 reports previously published lower bounds, corresponding to the respective maximum clique numbers.

Table 3. Lower bounds $\lceil \chi_f(G[X]) \rceil$ from induced subgraphs

Instance	$\lvert X \rvert$	$\lvert E(G[X]) \rvert$	$\lceil \chi_f(G[X]) \rceil$	old LB	UB	Time
`DSJC500.1`	300	5436	**9**	6	12	16 days
`DSJC1000.1`	350	8077	**10**	6	20	$<$ 36 days
`C2000.5`	1400	502370	**99**	16	148	$<$ 24 days
`C4000.5`	1500	589939	**107**	17	271	$<$ 26 days
`wap03a`	2500	164008	40	40	47	$<$ 3 days
`wap04a`	2500	159935	40	40	44	$<$ 1 days

5.4 Maximum-Weight Stable Set Results

Finally, we demonstrate the efficiency of Algorithm 1 for solving maximum-weight stable set problems. We compared the new algorithm with the currently fastest integer-programming solvers Gurobi 3.0.0 and CPLEX 12.2, as well as CLIQUER 1.21 [24], which solved the maximum-weight clique problems in the complement graphs. Where available, we used the final maximum-weight stable set instances as they occure in Algorithm 2. They can be downloaded from `http://code.google.com/p/exactcolors/wiki/MWISInstances`. Table 4 shows the results on the `DSJC*` instances and `C2000.5.1029`, to which we restrict ourselves here for space reasons. Comprehensive test results will be reported in the full version of the paper.

We performed two experiments per instance and solver. First, in the columns labeled STD, we computed the maximum-weight stable set as is. Second, in the columns labeled LB, we used the solvers in the same setting as in Algorithm 2 with an initial lower bound of $LB = 1$. We gave a time limit of ten hours for each run. `C2000.5.1029` is the only instance with $\alpha_\pi(G) > 1$, here Algorithm 1 was the fastest to find such a set. Algorithm 1 is competetive throughout all edge-density classes and was a main ingredient for improving known lower bounds.

Table 4. Running times of various MWIS solvers on hard instances in seconds

Instance	$\rho(G)$	Gurobi 3.0.0		CPLEX 12.2		CLIQUER 1.2		Algorithm 1	
	in %	STD	LB	STD	LB	STD	LB	STD	LB
`C2000.5.1029`	50	***	***	***	***	***	30586	***	11373
`DSJC250.1`	10.3	31278	34901	***	16288	***	***	5941	2281
`DSJC250.5`	50.3	1825	1963	2737	2557	1	1	1	1
`DSJC250.9`	89.6	1382	1442	319	317	1	1	1	1
`DSJC500.5`	50.1	***	***	***	***	9	9	32	32
`DSJC500.9`	90.1	***	***	24318	22105	1	1	1	1
`DSJC1000.5`	50	***	***	***	***	1076	1057	3634	3547
`DSJC1000.9`	89.9	***	***	***	***	1	1	2	2

Acknowledgments

We thank Andrew King for discussions on techniques to obtain dense subgraphs in large test instances.

References

1. Andrade, D.V., Resende, M.G.C., Werneck, R.F.: Fast local search for the maximum independent set problem. In: Proceedings of Workshop on Experimental Algorithms, pp. 220–234 (2008)
2. Applegate, D.L., Cook, W., Dash, S., Espinoza, D.G.: Exact solutions to linear programming problems. Operations Research Letters 35(6), 693–699 (2007)
3. Babel, L.: A fast algorithm for the maximum weight clique problem. Computing 52, 31–38 (1994)
4. Balas, E., Xue, J.: Minimum weighted coloring of triangulated graphs with application to maximum weight vertex packing and clique finding in arbitrary graphs. SIAM Journal of Computing 20(2), 209–221 (1991)
5. Balas, E., Yu, C.S.: Finding a maximum clique in an arbitrary graph. SIAM Journal of Computing 15(4), 1054–1068 (1986)
6. Brélaz, D.: New methods to color the vertices of a graph. Communications of the ACM 22(4), 251–256 (1979)
7. Caramia, M., Dell'Olmo, P.: Bounding vertex coloring by truncated multistage branch and bound. Networks 44(4), 231–242 (2004)
8. Gualandi, S., Malucelli, F.: Exact solution of graph coloring problems via constraint programming and column generation. Optimization Online in press for INFORMS Journal on Computing (2010)

9. Held, S., Sewell, E.C., Cook, W.: Exact colors project webpage (2010), http://code.google.com/p/exactcolors/
10. Lund, C., Yannakakis, M.: On the hardness of approximating minimization problems. Journal of the ACM 41(5), 960–981 (1994)
11. Malaguti, E., Monaci, M., Toth, P.: An exact approach for the vertex coloring problem. Discrete Optimization (2010) (in press)
12. Malaguti, E., Toth, P.: A survey on vertex coloring problems. International Transactions in Operational Research 17, 1–34 (2010)
13. Mehrotra, A., Trick, M.A.: A Column Generation Approach for Graph Coloring. INFORMS Journal on Computing 8(4), 344–354 (1996)
14. Mycielski, J.: Sur le coloriage des graphes. Colloq. Math. 3, 161–162 (1955)
15. Méndez-Díaz, I., Zabala, P.: A branch-and-cut algorithm for graph coloring. Discrete Applied Mathematics 154(5), 826–847 (2006)
16. Méndez Díaz, I., Zabala, P.: A cutting plane algorithm for graph coloring. Discrete Applied Mathematics 156(2), 159–179 (2008)
17. Porumbel, D.C., Hao, J.-K., Kuntz, P.: An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. Computers and Operations Research 37(10), 1822–1832 (2010)
18. Sewell, E.C.: A branch and bound algorithm for the stability number of a sparse graph. INFORMS Journal on Computing 10(4), 438–447 (1998)
19. Titiloye, O., Crispin, A.: Quantum annealing of the graph coloring problem. Discrete Optimization (2011) (in Press)
20. Trick, M.A.: DIMACS Graph Coloring Instances (2002), http://mat.gsia.cmu.edu/COLOR02/
21. Warren, J.S., Hicks, I.V.: Combinatorial branch-and-bound for the maximum weight independent set problem. Technical report, Texas A&M University (2006)
22. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. Theory of Computing 3(1), 103–128 (2007)
23. Östergård, P.R.J.: A new algorithm for the maximum-weight clique problem. Electronic Notes in Discrete Mathematics 3, 153–156 (1999)
24. Östergård, P.R.J., Niskanen, S.: Cliquer home page (2010), http://users.tkk.fi/pat/cliquer.html

Computing the Maximum Degree of Minors in Mixed Polynomial Matrices via Combinatorial Relaxation

Satoru Iwata[1,⋆] and Mizuyo Takamatsu[2,⋆]

[1] Research Institute for Mathematical Sciences
Kyoto University, Kyoto 606-8502, Japan
iwata@kurims.kyoto-u.ac.jp
[2] Department of Information and System Engineering
Chuo University, Tokyo 112-8551, Japan
takamatsu@ise.chuo-u.ac.jp

Abstract. Mixed polynomial matrices are polynomial matrices with two kinds of nonzero coefficients: fixed constants that account for conservation laws and independent parameters that represent physical characteristics. The computation of their maximum degrees of minors is known to be reducible to valuated independent assignment problems, which can be solved by polynomial numbers of additions, subtractions, and multiplications of rational functions. However, these arithmetic operations on rational functions are much more expensive than those on constants.

In this paper, we present a new algorithm of combinatorial relaxation type. The algorithm finds a combinatorial estimate of the maximum degree by solving a weighted bipartite matching problem, and checks if the estimate is equal to the true value by solving independent matching problems. The algorithm mainly relies on fast combinatorial algorithms and performs numerical computation only when necessary. In addition, it requires no arithmetic operations on rational functions. As a byproduct, this method yields a new algorithm for solving a linear valuated independent assignment problem.

1 Introduction

Let $A(s) = (A_{ij}(s))$ be a rational function matrix with $A_{ij}(s)$ being a rational function in s. The maximum degree $\delta_k(A)$ of minors of order k is defined by

$$\delta_k(A) = \max\{\deg \det A[I, J] \mid |I| = |J| = k\},$$

where $\deg z$ denotes the degree of a rational function $z(s)$, and $A[I, J]$ denotes the submatrix with row set I and column set J. This $\delta_k(A)$ determines the Smith-McMillan form at infinity of a rational function matrix [25], which is used in decoupling and disturbance rejection of linear time-invariant systems, and the

⋆ Supported by a Grant-in-Aid for Scientific Research from the Japan Society for Promotion of Science.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 274–286, 2011.

Kronecker canonical form of a matrix pencil [5, 24], which is used in analysis of linear DAEs with constant coefficients. Thus, the computation of $\delta_k(A)$ is a fundamental and important procedure in dynamical systems analysis.

The notion of mixed polynomial matrices [13, 20] was introduced as a mathematical tool for faithful description of dynamical systems such as electric circuits, mechanical systems, and chemical plants. A mixed polynomial matrix is a polynomial matrix that consists of two kinds of coefficients as follows.

Accurate Numbers (Fixed Constants). Numbers that account for conservation laws are precise in values. These numbers should be treated numerically.

Inaccurate Numbers (Independent Parameters). Numbers that represent physical characteristics are not precise in values. These numbers should be treated combinatorially as nonzero parameters without reference to their nominal values. Since each such nonzero entry often comes from a single physical device, the parameters are assumed to be independent.

For example, physical characteristics in engineering systems are not precise in values because of measurement noise, while exact numbers do arise in conservation laws such as Kirchhoff's conservation laws in electric circuits, or the law of conservation of mass, energy, or momentum and the principle of action and reaction in mechanical systems. Thus, it is natural to distinguish inaccurate numbers from accurate numbers in the description of dynamical systems.

In [19], Murota reduces the computation of $\delta_k(A)$ for a mixed polynomial matrix $A(s)$ to solving a *valuated independent assignment problem*, for which he presents in [17, 18] algorithms that perform polynomial numbers of additions, subtractions, and multiplications of rational functions. However, these arithmetic operations on rational functions are much more expensive than those on constants.

In this paper, we present an algorithm for computing $\delta_k(A)$, based on the framework of "combinatorial relaxation." The outline of the algorithm is as follows.

Phase 1. Construct a relaxed problem by discarding numerical information and extracting zero/nonzero structure in $A(s)$. The solution is regarded as an estimate of $\delta_k(A)$.

Phase 2. Check whether the obtained estimate is equal to the true value of $\delta_k(A)$, or not. If it is, return the estimate and halt.

Phase 3. Modify the relaxation so that the invalid solution is eliminated, and find a solution to the modified relaxed problem. Then go back to Phase 2.

In this algorithm, we solve a weighted bipartite matching problem in Phase 1 and independent matching problems in Phase 2. We remark that our algorithm does not need symbolic operations on rational functions.

This framework of combinatorial relaxation algorithm is somewhat analogous to the idea of relaxation and cutting plane in integer programming. In contrast to integer programming, where hard combinatorial problems are relaxed to linear

programs, here we relax a linear algebraic problem to an efficiently solvable combinatorial problem. Then the algorithm checks the validity of the obtained solution and modifies the relaxation if necessary just like adding a cutting plane. This is where the name "combinatorial relaxation" comes from.

We now summarize previous works based on the framework of combinatorial relaxation. The combinatorial relaxation approach was invented by Murota [14] for computing the Newton polygon of the Puiseux series solutions to determinantal equations. This approach was further applied to the computation of the degree of $\det A(s)$ in [16] and $\delta_k(A)$ in [7–9, 15] for a polynomial matrix $A(s)$. In computational efficiency of the algorithms, it is crucial to solve the relaxed problem efficiently and not to invoke the modification of the relaxation many times. The result in [9] shows practical efficiency of the combinatorial relaxation through computational experiments.

Let us have a closer look at the algorithms [7–9, 15] for $\delta_k(A)$. In [15], Murota presented a combinatorial relaxation algorithm for general rational function matrices using biproper equivalence transformations in the modification of the relaxed problem. The primal-dual version was presented in [9]. Another algorithm given in [8] solves a valuated independent assignment problem as a relaxed problem by regarding a given polynomial matrix as a mixed polynomial matrix. These algorithms need to transform a polynomial matrix by row/column operations, which possibly increase the number of terms in some entries. To avoid this phenomenon in the case of matrix pencils, Iwata [7] presented another combinatorial relaxation algorithm, which uses only strict equivalence transformations.

In this paper, we extend the combinatorial relaxation framework to mixed polynomial matrices. Our algorithm adopts a different way of matrix modification from the previous algorithms [8, 9, 15], which enables us to evaluate the complexity by the number of basic arithmetic operations. For an $m \times n$ mixed polynomial matrix with $m \leq n$, the algorithm runs in $O(m^{\omega+1}nd_{\max}^2)$ time, where ω is the matrix multiplication exponent and $d_{\max}$ is the maximum degree of an entry.

We compare this time complexity with that of the previous algorithm of Murota [19] based on the valuated independent assignment. The bottleneck in that algorithm is to transform an $m \times n$ polynomial matrix into an upper triangular matrix in each iteration. This can be done in $O^{\sim}(m^{\omega}nd_{\max})$ time, where $O^{\sim}$ indicates that we ignore $\log(md_{\max})$ factors, by using Bareiss' fraction-free Gaussian elimination approach [1, 2, 23] and an $O(d \log d \log \log d)$ time algorithm in [3] for multiplying polynomials of degree d. Since the number of iterations is k, Murota's algorithm runs in $O^{\sim}(km^{\omega}nd_{\max})$ time. Thus the worst-case complexity of our algorithm is comparable to that of the previous one.

However, our combinatorial relaxation algorithm terminates without invoking the modification of the relaxation unless there is an unlucky numerical cancellation. Consequently, in most cases, it runs in $O(m^{\omega}nd_{\max})$ time, which is much faster than the previous algorithm.

One application of our combinatorial relaxation algorithm is to compute the Kronecker canonical form of a *mixed matrix pencil*, which is a mixed polynomial

matrix with $d_{\max} = 1$. For an $n \times n$ regular mixed matrix pencil, our algorithm enables us to compute the Kronecker canonical form in $O(n^{\omega+2})$ time. This time complexity is better than the previous algorithm given in [10], which makes use of another characterization based on expanded matrices.

Another application is to compute the optimal value of a linear valuated independent assignment problem. Since the optimal value coincides with the degree of the determinant of the associated mixed polynomial matrix, we can make use of our combinatorial relaxation algorithm. This means that we find the optimal value of a linear valuated independent assignment problem by solving a sequence of independent matching problems.

Combinatorial relaxation approach exploits the combinatorial structures of polynomial matrices and exhibits a connection between combinatorial optimization and matrix computation. While recent works of Mucha and Sankowski [11, 12], Sankowski [22], and Harvey [6] utilize matrix computation to solve matching problems, this paper adopts the opposite direction, that is, we utilize matching problems for matrix computation.

The organization of this paper is as follows. Section 2 is devoted to preliminaries on rational function matrices, the independent matching problem, and mixed matrix theory. We present a combinatorial relaxation algorithm in Section 3, and analyze its complexity in Section 4. In Section 5, we apply the combinatorial relaxation approach to the linear valuated independent assignment problem.

2 Preliminaries

2.1 Rational Function Matrices

We denote the degree of a polynomial $z(s)$ by $\deg z$, where $\deg 0 = -\infty$ by convention. For a rational function $f(s) = g(s)/h(s)$ with polynomials $g(s)$ and $h(s)$, its degree is defined by $\deg f(s) = \deg g(s) - \deg h(s)$. A rational function $f(s)$ is called *proper* if $\deg f(s) \leq 0$, and *strictly proper* if $\deg f(s) < 0$. We call a rational function matrix (*strictly*) *proper* if its entries are (strictly) proper rational functions. A square proper rational function matrix is called *biproper* if it is invertible and its inverse is a proper rational function matrix. A proper rational function matrix is biproper if and only if its determinant is a nonzero constant. It is known that $\delta_k(A)$ is invariant under biproper equivalence transformations, i.e., $\delta_k(A) = \delta_k(\tilde{A})$ for $k = 1, \ldots, \operatorname{rank} A(s)$ if $\tilde{A}(s) = U(s)A(s)V(s)$ with biproper matrices $U(s)$ and $V(s)$.

A rational function matrix $Z(s)$ is called a *Laurent polynomial matrix* if $s^N Z(s)$ is a polynomial matrix for some integer N. In our algorithm, we make use of biproper Laurent polynomial matrices in the phase of matrix modification.

2.2 Independent Matching Problem

A *matroid* is a pair $\mathbf{M} = (V, \mathcal{I})$ of a finite set V and a collection $\mathcal{I}$ of subsets of V such that

(I-1) $\emptyset \in \mathcal{I}$,
(I-2) $I \subseteq J \in \mathcal{I} \Rightarrow I \in \mathcal{I}$,
(I-3) $I, J \in \mathcal{I}, |I| < |J| \Rightarrow I \cup \{v\} \in \mathcal{I}$ for some $v \in J \setminus I$.

The set V is called the *ground set*, $I \in \mathcal{I}$ is an *independent set*, and $\mathcal{I}$ is the *family of independent sets*. The following problem is an extension of the matching problem.

[Independent Matching Problem (IMP)]
Given a bipartite graph $G = (V^+, V^-; E)$ with vertex sets V^+, V^- and edge set E, and a pair of matroids $\mathbf{M}^+ = (V^+, \mathcal{I}^+)$ and $\mathbf{M}^- = (V^-, \mathcal{I}^-)$, find a matching $M \subseteq E$ that maximizes $|M|$ subject to $\partial^+ M \in \mathcal{I}^+$ and $\partial^- M \in \mathcal{I}^-$, where $\partial^+ M$ and $\partial^- M$ denote the set of vertices in V^+ and V^- incident to M, respectively.

A matching $M \subseteq E$ satisfying $\partial^+ M \in \mathcal{I}^+$ and $\partial^- M \in \mathcal{I}^-$ is called an *independent matching*.

2.3 Mixed Matrices and Mixed Polynomial Matrices

A *generic matrix* is a matrix in which each nonzero entry is an independent parameter. A matrix A is called a *mixed matrix* if A is given by $A = Q + T$ with a constant matrix Q and a generic matrix T. A *layered mixed matrix* (or an *LM-matrix* for short) is defined to be a mixed matrix such that Q and T have disjoint nonzero rows. An LM-matrix A is expressed by $A = \binom{Q}{T}$.

A polynomial matrix $A(s)$ is called a *mixed polynomial matrix* if $A(s)$ is given by $A(s) = Q(s) + T(s)$ with a pair of polynomial matrices $Q(s) = \sum_{k=0}^{N} s^k Q_k$ and $T(s) = \sum_{k=0}^{N} s^k T_k$ that satisfy the following two conditions.

(MP-Q) Q_k $(k = 0, 1, \ldots, N)$ are constant matrices.
(MP-T) T_k $(k = 0, 1, \ldots, N)$ are generic matrices.

A *layered mixed polynomial matrix* (or an *LM-polynomial matrix* for short) is defined to be a mixed polynomial matrix such that $Q(s)$ and $T(s)$ have disjoint nonzero rows. An LM-polynomial matrix $A(s)$ is expressed by $A(s) = \binom{Q(s)}{T(s)}$. If $Q(s)$ and $T(s)$ are Laurent polynomial matrices, we call $A(s) = \binom{Q(s)}{T(s)}$ an *LM-Laurent polynomial matrix*. For $A(s) = \binom{Q(s)}{T(s)}$, we denote the row set and the column set of $A(s)$ by R and C, and the row sets of $Q(s)$ and $T(s)$ by R_Q and R_T. We also set $m_Q = |R_Q|$, $m_T = |R_T|$, and $n = |C|$ for convenience. The (i, j) entry of $Q(s)$ and $T(s)$ is denoted by $Q_{ij}(s)$ and $T_{ij}(s)$, respectively. We use these notations throughout this paper for LM-Laurent polynomial matrices as well as LM-matrices. For an LM-Laurent polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$, let us define

$$\delta_k^{\mathrm{LM}}(A) = \{\deg \det A[R_Q \cup I, J] \mid I \subseteq R_T, J \subseteq C, |I| = k, |J| = m_Q + k\},$$

where $0 \leq k \leq \min(m_T, n - m_Q)$. Note that $\delta_k^{\mathrm{LM}}(A)$ designates the maximum degree of minors of order $m_Q + k$ with row set containing R_Q.

We denote an $m \times m$ diagonal matrix with the (i,i) entry being a_i by $\mathrm{diag}[a_1, \ldots, a_m]$. Let $\tilde{A}(s) = \tilde{Q}(s) + \tilde{T}(s)$ be an $m \times n$ mixed polynomial matrix with row set R and column set C. We construct a $(2m) \times (m+n)$ LM-polynomial matrix

$$A(s) = \begin{pmatrix} \mathrm{diag}[s^{d_1}, \ldots, s^{d_m}] & \tilde{Q}(s) \\ -\mathrm{diag}[t_1 s^{d_1}, \ldots, t_m s^{d_m}] & \tilde{T}(s) \end{pmatrix}, \tag{1}$$

where t_i $(i = 1, \ldots, m)$ are independent parameters and $d_i = \max_{j \in C} \deg \tilde{Q}_{ij}(s)$ for $i \in R$. A mixed polynomial matrix $\tilde{A}(s)$ and its associated LM-polynomial matrix $A(s)$ have the following relation, which implies that the value of $\delta_k(\tilde{A})$ is obtained from $\delta_k^{\mathrm{LM}}(A)$.

Lemma 1 ([20, Lemma 6.2.6]). *Let $\tilde{A}(s)$ be an $m \times n$ mixed polynomial matrix and $A(s)$ the associated LM-polynomial matrix defined by (1). Then it holds that $\delta_k(\tilde{A}) = \delta_k^{\mathrm{LM}}(A) - \sum_{i=1}^{m} d_i$.*

2.4 Rank of LM-Matrices

The computation of the rank of an LM-matrix $A = \binom{Q}{T}$ is reduced to solving an independent matching problem [21] as follows. See [20, §4.2.4] for details.

Let $C_Q = \{j_Q \mid j \in C\}$ be a copy of the column set C of A. Consider a bipartite graph $G = (V^+, V^-; E_T \cup E_Q)$ with $V^+ = R_T \cup C_Q$, $V^- = C$,

$$E_T = \{(i,j) \mid i \in R_T, j \in C, T_{ij} \neq 0\} \quad \text{and} \quad E_Q = \{(j_Q, j) \mid j \in C\}.$$

Let $\mathbf{M}^+ = (V^+, \mathcal{I}^+)$ be a matroid defined by

$$\mathcal{I}^+ = \{I^+ \subseteq V^+ \mid \mathrm{rank}\, Q[R_Q, I^+ \cap C_Q] = |I^+ \cap C_Q|\},$$

and $\mathbf{M}^-$ be a free matroid. We consider the independent matching problem with respect to G, $\mathbf{M}^+$, and $\mathbf{M}^-$. Then $\mathrm{rank}\, A$ has the following property.

Theorem 1 ([20, Theorem 4.2.18]). *Let A be an LM-matrix. Then the rank of A is equal to the maximum size of an independent matching in the problem defined above, i.e.,* $\mathrm{rank}\, A = \max\{|M| \mid M : \textit{independent matching}\}$.

Let $\tilde{A} = \binom{\tilde{Q}}{\tilde{T}}$ be an LM-matrix obtained at the termination of the algorithm for solving the independent matching problem. Then we have $\tilde{A} = \begin{pmatrix} U & O \\ O & I \end{pmatrix} \begin{pmatrix} Q \\ T \end{pmatrix}$ for some nonsingular constant matrix U.

The structure of an LM-matrix $\tilde{A} = (\tilde{A}_{ij})$ is represented by a bipartite graph $G(\tilde{A}) = (R_Q \cup R_T, C; E(\tilde{A}))$ with $E(\tilde{A}) = \{(i,j) \mid i \in R_Q \cup R_T, j \in C, \tilde{A}_{ij} \neq 0\}$. The maximum size of a matching in $G(\tilde{A})$ is called the *term-rank* of $\tilde{A}$, denoted by $\text{term-rank}\, \tilde{A}$. Now $\tilde{A}$ has the following property.

Lemma 2. *Let $\tilde{A}$ be an LM-matrix obtained at the termination of the algorithm for solving the independent matching problem with respect to G, $\mathbf{M}^+$, and $\mathbf{M}^-$. Then $\tilde{A}$ satisfies* $\mathrm{rank}\, \tilde{A} = \text{term-rank}\, \tilde{A}$.

3 Combinatorial Relaxation Algorithm

In this section, we present a combinatorial relaxation algorithm to compute $\delta_k^{\mathrm{LM}}(A)$ for an LM-polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$. We assume that $Q(s)$ is of full row rank. Note that this assumption is valid for the associated LM-polynomial matrix defined by (1). Since an LM-polynomial matrix is transformed into an LM-Laurent polynomial matrix in the phase of matrix modification, we hereafter deal with an LM-Laurent polynomial matrix.

3.1 Combinatorial Relaxation

Let us construct a bipartite graph $G(A) = (R_Q \cup R_T, C; E(A))$ with $E(A) = \{(i,j) \mid i \in R_Q \cup R_T, j \in C, A_{ij}(s) \neq 0\}$. The weight $c(e)$ of an edge $e = (i,j)$ is given by $c(e) = c_{ij} = \deg A_{ij}(s)$. We remark that $c(e)$ is integer for any $e \in E(A)$ if $A(s)$ is an LM-Laurent polynomial matrix. The maximum weight of a matching in $G(A)$ denoted by $\hat{\delta}_k^{\mathrm{LM}}(A)$ is an upper bound on $\delta_k^{\mathrm{LM}}(A)$. We adopt $\hat{\delta}_k^{\mathrm{LM}}(A)$ as an estimate of $\delta_k^{\mathrm{LM}}(A)$.

Consider the following linear program (PLP(A,k)):

$$\begin{aligned}
\text{maximize} \quad & \sum_{e \in E(A)} c(e)\xi(e) \\
\text{subject to} \quad & \sum_{\partial e \ni i} \xi(e) = 1 \quad (\forall i \in R_Q), \\
& \sum_{\partial e \ni i} \xi(e) \leq 1 \quad (\forall i \in R_T \cup C), \\
& \sum_{e \in E(A)} \xi(e) = m_Q + k, \\
& \xi(e) \geq 0 \quad (\forall e \in E(A)),
\end{aligned}$$

where ∂e denotes the set of vertices incident to e. The first constraint represents that M must satisfy $\partial M \supseteq R_Q$, where ∂M denotes the vertices incident to edges in M. By the total unimodularity of the coefficient matrix, PLP(A,k) has an integral optimal solution with $\xi(e) \in \{0,1\}$ for any $e \in E(A)$. This optimal solution corresponds to the maximum weight matching M in $G(A)$, and its optimal value $c(M) = \sum_{e \in M} c(e)$ is equal to $\hat{\delta}_k^{\mathrm{LM}}(A)$. The dual program (DLP$(A,k)$) is expressed as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in R} p_i + \sum_{j \in C} q_j + (m_Q + k)t \\
\text{subject to} \quad & p_i + q_j + t \geq c(e) \quad (\forall e = (i,j) \in E(A)), \\
& p_i \geq 0 \quad (\forall i \in R_T), \\
& p_j \geq 0 \quad (\forall j \in C).
\end{aligned}$$

Then DLP(A,k) has an integral optimal solution, because the coefficient matrix is totally unimodular and $c(e)$ is integer for any $e \in E(A)$.

The outline of the combinatorial relaxation algorithm to compute $\delta_k^{\mathrm{LM}}(A)$ is summarized as follows.

Outline of Algorithm for Computing $\delta_k^{\mathrm{LM}}(A)$

Phase 1. Find a maximum weight matching M such that $\partial M \supseteq R_Q$ and $|M| = m_Q + k$ in $G(A)$. Then the maximum weight $\hat{\delta}_k^{\mathrm{LM}}(A)$ is regarded as an estimate of $\delta_k^{\mathrm{LM}}(A)$. Construct an optimal solution (p, q, t) of $\mathrm{DLP}(A, k)$ from M.

Phase 2. Test whether $\hat{\delta}_k^{\mathrm{LM}}(A) = \delta_k^{\mathrm{LM}}(A)$ or not by using (p, q, t). If equality holds, return $\hat{\delta}_k^{\mathrm{LM}}(A)$ and halt.

Phase 3. Modify $A(s)$ to another matrix $\tilde{A}(s)$ such that $\hat{\delta}_k^{\mathrm{LM}}(\tilde{A}) \leq \hat{\delta}_k^{\mathrm{LM}}(A) - 1$ and $\delta_k^{\mathrm{LM}}(\tilde{A}) = \delta_k^{\mathrm{LM}}(A)$. Construct an optimal solution $(\tilde{p}, \tilde{q}, \tilde{t})$ of $\mathrm{DLP}(\tilde{A}, k)$ from (p, q, t), and go back to Phase 2.

In Phase 1, we find a maximum weight matching M by using an efficient combinatorial algorithm. An optimal solution (p, q, t) can be obtained by using shortest path distance in an associated auxiliary graph. In Phase 2, we check whether an upper estimate $\hat{\delta}_k^{\mathrm{LM}}(A)$ coincides with $\delta_k^{\mathrm{LM}}(A)$ by computing the ranks of LM-matrices. If it does not, we transform $A(s)$ into $\tilde{A}(s)$ so that the estimate $\hat{\delta}_k^{\mathrm{LM}}(A)$ is eliminated. We repeat this procedure until the upper bound coincides with $\delta_k^{\mathrm{LM}}(A)$. Since an upper estimate decreases at each step, the iteration terminates at most $\hat{\delta}_k^{\mathrm{LM}}(A)$ times. We explain in detail Phases 2 in Section 3.2, and Phase 3 in Sections 3.3 and 3.4.

3.2 Test for Tightness

We describe a necessary and sufficient condition for $\hat{\delta}_k^{\mathrm{LM}}(A) = \delta_k^{\mathrm{LM}}(A)$. For an integral feasible solution (p, q, t) of $\mathrm{DLP}(A, k)$, let us put $I^* = R_Q \cup \{i \in R_T \mid p_i > 0\}$ and $J^* = \{j \in C \mid q_j > 0\}$. We call I^* and J^* *active rows* and *active columns*, respectively. The *tight coefficient matrix* $A^* = (A^*_{ij})$ is defined by $A^*_{ij} = (\text{the coefficient of } s^{p_i+q_j+t} \text{ in } A_{ij}(s))$. Note that A^* is an LM-matrix and $A(s) = (A_{ij}(s))$ is expressed by

$$A_{ij}(s) = s^{p_i+q_j+t}(A^*_{ij} + A^\infty_{ij}) \tag{2}$$

with a strictly proper matrix $A^\infty = (A^\infty_{ij})$.

The following lemma gives a necessary and sufficient condition for $\hat{\delta}_k^{\mathrm{LM}}(A) = \delta_k^{\mathrm{LM}}(A)$, which is immediately derived from [15, Theorem 7].

Lemma 3. *Let (p, q, t) be an optimal dual solution, I^* and J^* the active rows and columns, and A^* the tight coefficient matrix. Then $\hat{\delta}_k^{\mathrm{LM}}(A) = \delta_k^{\mathrm{LM}}(A)$ if and only if the following four conditions are satisfied:*

(r1) $\operatorname{rank} A^*[R, C] \geq m_Q + k$,
(r2) $\operatorname{rank} A^*[I^*, C] = |I^*|$,
(r3) $\operatorname{rank} A^*[R, J^*] = |J^*|$,
(r4) $\operatorname{rank} A^*[I^*, J^*] \geq |I^*| + |J^*| - (m_Q + k)$.

Lemma 3 implies that we can check $\hat{\delta}_k^{\mathrm{LM}}(A) = \delta_k^{\mathrm{LM}}(A)$ efficiently by computing the ranks of four LM-matrices $A^*[R, C]$, $A^*[I^*, C]$, $A^*[R, J^*]$, and $A^*[I^*, J^*]$. This can be done by solving the corresponding independent matching problems. The optimality condition for (p, q, t) is given by the following variant, which is also derived from [15].

Lemma 4. *Let (p, q, t) be a dual feasible solution, I^* and J^* the active rows and columns, and A^* the tight coefficient matrix. Then (p, q, t) is optimal if and only if the following four conditions are satisfied:*

(t1) term-rank $A^*[R, C] \geq m_Q + k$,
(t2) term-rank $A^*[I^*, C] = |I^*|$,
(t3) term-rank $A^*[R, J^*] = |J^*|$,
(t4) term-rank $A^*[I^*, J^*] \geq |I^*| + |J^*| - (m_Q + k)$.

3.3 Matrix Modification

Let $A(s)$ be an LM-Laurent polynomial matrix such that $\delta_k^{\mathrm{LM}}(A) < \hat{\delta}_k^{\mathrm{LM}}(A)$. We describe the rule of modifying $A(s)$ into another LM-Laurent polynomial matrix $\tilde{A}(s)$. Since $\delta_k^{\mathrm{LM}}(A) < \hat{\delta}_k^{\mathrm{LM}}(A)$, it follows from Lemma 3 that at least one of conditions (r1)–(r4) is violated. In the test for tightness in Phase 2, we transform the tight coefficient matrix $A^* = \binom{Q^*}{T^*}$ into another LM-matrix $\tilde{A}^* = \binom{\tilde{Q}^*}{\tilde{T}^*}$ by solving the independent matching problem described in Section 2.4. Then, we find a nonsingular constant matrix U such that $\begin{pmatrix} \tilde{Q}^* \\ \tilde{T}^* \end{pmatrix} = \begin{pmatrix} U & O \\ O & I \end{pmatrix} \begin{pmatrix} Q^* \\ T^* \end{pmatrix}$.

Let (p, q, t) be an optimal solution of DLP(A, k). We transform $A(s)$ into another LM-Laurent polynomial matrix $\tilde{A}(s)$ defined by

$$\tilde{A}(s) = \mathrm{diag}[s^{p_1}, \ldots, s^{p_m}] \begin{pmatrix} U & O \\ O & I \end{pmatrix} \mathrm{diag}[s^{-p_1}, \ldots, s^{-p_m}] A(s). \tag{3}$$

It is ensured from [15, Lemma 11] that $\delta_k^{\mathrm{LM}}(A)$ is invariant under the transformation (3). In addition, it can be shown that an optimal solution (p, q, t) of DLP(A, k) is feasible for DLP$(\tilde{A}, k)$ but not optimal.

Lemma 5. *Let $A(s)$ be an LM-Laurent polynomial matrix satisfying $\delta_k^{\mathrm{LM}}(A) < \hat{\delta}_k^{\mathrm{LM}}(A)$, and $\tilde{A}(s)$ the LM-Laurent polynomial matrix defined by (3). Then an optimal solution (p, q, t) of DLP(A, k) is feasible for DLP$(\tilde{A}, k)$ but not optimal.*

Proof. We put $F(s) = s^{-t} \mathrm{diag}[s^{-p_1}, \ldots, s^{-p_m}] \tilde{A}(s) \mathrm{diag}[s^{-q_1}, \ldots, s^{-q_n}]$. Then $\deg F_{ij}(s) = \tilde{c}_{ij} - p_i - q_j - t$ holds, where $\tilde{c}_{ij} = \deg \tilde{A}_{ij}(s)$. It follows from (2) and (3) that $F(s) = \tilde{U}(A^* + A^\infty)$ with $\tilde{U} = \begin{pmatrix} U & O \\ O & I \end{pmatrix}$ and a strictly proper matrix A^∞. Since U and A^* are constant matrices, we have $\deg F_{ij}(s) \leq 0$, which implies that (p, q, t) is feasible for DLP$(\tilde{A}, k)$.

We show that (p, q, t) is not optimal for DLP$(\tilde{A}, k)$. The matrix $\tilde{A}^*_{(p,q,t)} := \tilde{U}A^*$ is the tight coefficient matrix of $\tilde{A}(s)$ with respect to (p, q, t). By $\delta^{\mathrm{LM}}_k(A) < \hat{\delta}^{\mathrm{LM}}_k(A)$ and Lemma 3, at least one of conditions (r1)–(r4) is violated. Let us assume that A^* violates (r2) and $\tilde{A}^*_{(p,q,t)}$ is obtained by computing rank $A^*[I^*, C]$. Then we have term-rank $\tilde{A}^*_{(p,q,t)}[I^*, C] = \operatorname{rank} \tilde{A}^*_{(p,q,t)}[I^*, C] = \operatorname{rank} A^*[I^*, C] < |I^*|$ by Lemma 2. Thus $\tilde{A}^*_{(p,q,t)}$ violates (t2), which implies that (p, q, t) is not optimal for DLP$(\tilde{A}, k)$ by Lemma 4. If (r1), (r3), or (r4) is violated, we can prove that (p, q, t) is not optimal for DLP$(\tilde{A}, k)$ in a similar way.

3.4 Dual Updates

Let (p, q, t) be an optimal solution of DLP(A, k). By Lemma 5, (p, q, t) is feasible for DLP$(\tilde{A}, k)$. For (p, q, t) and another feasible solution (p', q', t') of DLP$(\tilde{A}, k)$, we consider the amount of change Δ in the value of the dual objective function defined by

$$\Delta = \left(\sum_{i \in R} p'_i + \sum_{j \in C} q'_j + (m_Q + k)t' \right) - \left(\sum_{i \in R} p_i + \sum_{j \in C} q_j + (m_Q + k)t \right).$$

With the use of (p, q, t), we construct a feasible solution (p', q', t') which satisfies $\Delta < 0$. By repeating this procedure, we find an optimal dual solution of DLP$(\tilde{A}, k)$.

Let $G^* = (R, C; E^*)$ be a bipartite graph with $E^* = \{(i, j) \in E(\tilde{A}) \mid p_i + q_j + t = \tilde{c}_{ij}\}$, where $\tilde{c}_{ij} = \deg \tilde{A}_{ij}(s)$. Since (p, q, t) is not optimal for DLP$(\tilde{A}, k)$ by Lemma 5, at least one of conditions (t1)–(t4) for $\tilde{A}^*_{(p,q,t)}$ is violated. For each case, we construct another feasible solution (p', q', t') with $\Delta < 0$ as follows.

Case1: (t1) is Violated. Since the maximum size of a matching in $G^* = (R, C; E^*)$ is strictly less than $m_Q + k$, G^* has a cover U with $|U| < m_Q + k$. We now define $t' = t - 1$,

$$p'_i = \begin{cases} p_i + 1 & (i \in R \cap U) \\ p_i & (i \in R \setminus U) \end{cases}, \qquad q'_j = \begin{cases} q_j + 1 & (j \in C \cap U) \\ q_j & (j \in C \setminus U) \end{cases}.$$

Then it holds that $\Delta = |R \cap U| + |C \cap U| - (m_Q + k) = |U| - (m_Q + k) < 0$.

The resulting (p', q', t') is feasible for DLP$(\tilde{A}, k)$. In the other cases, (p', q', t') is given as follows.

Case2: (t2) is Violated. Let U be a cover of $G^*[I^* \cup C] = (I^*, C; E^*[I^* \cup C])$. We define $t' = t$,

$$p'_i = \begin{cases} p_i & (i \in (I^* \cap U) \cup (R \setminus I^*)) \\ p_i - 1 & (i \in I^* \setminus U) \end{cases}, \qquad q'_j = \begin{cases} q_j + 1 & (j \in C \cap U) \\ q_j & (j \in C \setminus U) \end{cases}.$$

Case3: (t3) is Violated. Let U be a cover of $G^*[R \cup J^*] = (R, J^*; E^*[R \cup J^*])$. We define $t' = t$,

$$p'_i = \begin{cases} p_i + 1 & (i \in R \cap U) \\ p_i & (i \in R \setminus U) \end{cases}, \qquad q'_j = \begin{cases} q_j & (j \in (J^* \cap U) \cup (C \setminus J^*)) \\ q_j - 1 & (j \in J^* \setminus U) \end{cases}.$$

Case4: (t4) is Violated. Let U be a cover of $G^*[I^* \cup J^*] = (I^*, J^*; E^*[I^* \cup J^*])$. We define $t' = t + 1$,

$$p'_i = \begin{cases} p_i - 1 & (i \in I^* \setminus U) \\ p_i & (i \in (I^* \cap U) \cup (R \setminus I^*)) \end{cases}, q'_j = \begin{cases} q_j - 1 & (j \in J^* \setminus U) \\ q_j & (j \in (J^* \cap U) \cup (C \setminus J^*)) \end{cases}.$$

4 Complexity Analysis

We now analyze the complexity of our combinatorial relaxation algorithm. The algorithm is dominated by the computation of $\tilde{A}(s)$ in the phase of matrix modification.

For an integral optimal solution (p, q, t) of DLP(A, k), let $P(s)$ and $Q(s)$ denote $\mathrm{diag}[s^{p_1}, \ldots, s^{p_m}]$ and $\mathrm{diag}[s^{q_1}, \ldots, s^{q_n}]$, respectively. By the definition of the tight coefficient matrix A^*, the LM-Laurent polynomial matrix $A(s)$ is expressed as $A(s) = s^t P(s) \left(A^* + \frac{1}{s} A_1 + \frac{1}{s^2} A_2 + \cdots + \frac{1}{s^l} A_l \right) Q(s)$ for some integer l, where A_i denotes a constant matrix for $i = 1, 2, \ldots l$. By (3), we have

$$\tilde{A}(s) = s^t P(s) \tilde{U} \left(A^* + \frac{1}{s} A_1 + \frac{1}{s^2} A_2 + \cdots + \frac{1}{s^l} A_l \right) Q(s),$$

where $\tilde{U} = \begin{pmatrix} U & O \\ O & I \end{pmatrix}$. Thus, it suffices to perform constant matrix multiplications $\tilde{U}A^*$, $\tilde{U}A_1, \ldots, \tilde{U}A_l$. The following lemma guarantees that we may assume that l is at most $\hat{\delta}_k^{\mathrm{LM}}(A)$.

Lemma 6. *Let $A(s) = \binom{Q(s)}{T(s)}$ be an LM-polynomial matrix such that $Q(s)$ is of full row rank, and (p, q, t) an optimal solution of DLP(A, k). Then $\delta_k^{\mathrm{LM}}(A) = \delta_k^{\mathrm{LM}}(\bar{A})$ holds for*

$$\bar{A}(s) = s^t P(s) \left(A^* + \frac{1}{s} A_1 + \frac{1}{s^2} A_2 + \cdots + \frac{1}{s^{\hat{\delta}_k^{\mathrm{LM}}(A)}} A_{\hat{\delta}_k^{\mathrm{LM}}(A)} \right) Q(s),$$

which is obtained by ignoring the terms $\frac{1}{s^i} A_i$ with $i > \hat{\delta}_k^{\mathrm{LM}}(A)$.

By Lemma 6, the time and space complexities of the algorithm are as follows.

Theorem 2. *Let $A(s) = \binom{Q(s)}{T(s)}$ be an $m \times n$ LM-polynomial matrix such that $Q(s)$ is of full row rank, and $d_{\max}$ the maximum degree of an entry in $A(s)$. Then the algorithm runs in $O((m_Q + k)^2 m_Q^{\omega-1} n d_{\max}^2)$ time and $O((m_Q + k) m n d_{\max})$ space, where $|R_Q| = m_Q$ and $\omega < 2.38$ is the matrix multiplication exponent.*

The number of matrix modifications is at most $\hat{\delta}_k^{\mathrm{LM}}(A) - \delta_k^{\mathrm{LM}}(A)$. In the proof of Theorem 2, this value is bounded by $O((m_Q + k)d_{\max})$. In most cases, however, the difference is so small that it can be regarded as a constant. Thus the algorithm effectively runs in $O((m_Q + k)m_Q^{\omega-1}nd_{\max})$ time, which is much faster than suggested by Theorem 2.

5 Application to Valuated Independent Assignment

As a generalization of matroids, Dress and Wenzel [4] introduced *valuated matroids*. A valuated matroid $\mathbf{M} = (V, \mathcal{B}, \omega)$ is a triple of a ground set V, a base family $\mathcal{B} \subseteq 2^V$, and a function $\omega : \mathcal{B} \to \mathbb{R}$ that satisfy the following axiom (VM).

(VM) For any $B, B' \in \mathcal{B}$ and $u \in B \setminus B'$, there exists $v \in B' \setminus B$ such that $B \setminus \{u\} \cup \{v\} \in \mathcal{B}$, $B' \cup \{u\} \setminus \{v\} \in \mathcal{B}$, and $\omega(B) + \omega(B') \leq \omega(B \setminus \{u\} \cup \{v\}) + \omega(B' \cup \{u\} \setminus \{v\})$.

The function ω is called a *valuation*.

Murota [17, 18] introduced the *valuated independent assignment problem* as a generalization of the independent matching problem.

> **[Valuated Independent Assignment Problem (VIAP)]**
> Given a bipartite graph $G = (V^+, V^-; E)$ with vertex sets V^+, V^- and edge set E, a pair of valuated matroids $\mathbf{M}^+ = (V^+, \mathcal{B}^+, \omega^+)$ and $\mathbf{M}^- = (V^-, \mathcal{B}^-, \omega^-)$, and a weight function $w : E \to \mathbb{R}$, find a matching $M \subseteq E$ that maximizes $\Omega(M) := w(M) + \omega^+(\partial^+ M) + \omega^-(\partial^- M)$ subject to $\partial^+ M \in \mathcal{B}^+$ and $\partial^- M \in \mathcal{B}^-$.

Let $\mathbf{M}^+$ and $\mathbf{M}^-$ be linear valuated matroids represented by polynomial matrices $Q_+(s)$ and $Q_-(s)$, respectively. For a bipartite graph $G = (V^+, V^-; E)$, let $T(s) = \sum_{k=0}^{N} s^k T_k$ be a polynomial matrix which satisfies (MP-T), $E = \{(i, j) \mid T_{ij}(s) \neq 0\}$, and $\deg T_{ij}(s) = w(e)$ for $e = (i, j) \in E$. Then the optimal value of $\Omega(M)$ is equal to the degree of the determinant of the mixed polynomial matrix

$$A(s) = \begin{pmatrix} O & Q_+(s)^\top & O \\ Q_-(s) & O & I \\ O & I & T(s) \end{pmatrix}.$$

We obtain $\deg \det A(s)$ by using our combinatorial relaxation algorithm. This means that we can find the optimal value of the linear valuated independent assignment problem by solving a sequence of independent matching problems.

References

1. Bareiss, E.H.: Sylvester's identity and multistep integer-preserving Gaussian elimination. Mathematics of Computation 22, 565–578 (1968)
2. Bareiss, E.H.: Computational solutions of matrix problems over an integral domain. IMA Journal of Applied Mathematics 10, 68–104 (1972)

3. Cantor, D.G., Kaltofen, E.: On fast multiplication of polynomials over arbitrary algebras. Acta Informatica 28, 693–701 (1991)
4. Dress, A.W.M., Wenzel, W.: Valuated matroids. Advances in Mathematics 93, 214–250 (1992)
5. Gantmacher, F.R.: The Theory of Matrices. Chelsea, New York (1959)
6. Harvey, N.J.A.: Algebraic algorithms for matching and matroid problems. SIAM Journal on Computing 39, 679–702 (2009)
7. Iwata, S.: Computing the maximum degree of minors in matrix pencils via combinatorial relaxation. Algorithmica 36, 331–341 (2003)
8. Iwata, S., Murota, K.: Combinatorial relaxation algorithm for mixed polynomial matrices. Mathematical Programming 90, 353–371 (2001)
9. Iwata, S., Murota, K., Sakuta, I.: Primal-dual combinatorial relaxation algorithms for the maximum degree of subdeterminants. SIAM Journal on Scientific Computing 17, 993–1012 (1996)
10. Iwata, S., Takamatsu, M.: On the Kronecker canonical form of mixed matrix pencils. SIAM Journal on Matrix Analysis and Applications 32, 44–71 (2011)
11. Mucha, M., Sankowski, P.: Maximum matchings via Gaussian elimination. In: 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 248–255. IEEE Computer Society, Los Alamitos (2004)
12. Mucha, M., Sankowski, P.: Maximum matchings in planar graphs via Gaussian elimination. Algorithmica 45, 3–20 (2006)
13. Murota, K.: Systems Analysis by Graphs and Matroids — Structural Solvability and Controllability. Springer, Berlin (1987)
14. Murota, K.: Computing Puiseux-series solutions to determinantal equations via combinatorial relaxation. SIAM Journal on Computing 19, 1132–1161 (1990)
15. Murota, K.: Combinatorial relaxation algorithm for the maximum degree of subdeterminants: Computing Smith-McMillan form at infinity and structural indices in Kronecker form. Applicable Algebra in Engineering, Communication and Computing 6, 251–273 (1995)
16. Murota, K.: Computing the degree of determinants via combinatorial relaxation. SIAM Journal on Computing 24, 765–796 (1995)
17. Murota, K.: Valuated matroid intersection, I: Optimality criteria. SIAM Journal on Discrete Mathematics 9, 545–561 (1996)
18. Murota, K.: Valuated matroid intersection, II: Algorithms. SIAM Journal on Discrete Mathematics 9, 562–576 (1996)
19. Murota, K.: On the degree of mixed polynomial matrices. SIAM Journal on Matrix Analysis and Applications 20, 196–227 (1999)
20. Murota, K.: Matrices and Matroids for Systems Analysis. Springer, Berlin (2000)
21. Murota, K., Iri, M.: Structural solvability of systems of equations — A mathematical formulation for distinguishing accurate and inaccurate numbers in structural analysis of systems. Japan Journal of Applied Mathematics 2, 247–271 (1985)
22. Sankowski, P.: Maximum weight bipartite matching in matrix multiplication time. Theoretical Computer Science 410, 4480–4488 (2009)
23. Storjohann, A.: Algorithms for Matrix Canonical Forms. Ph.D. thesis, ETH Zürich (2000)
24. Thorp, J.S.: The singular pencil of a linear dynamical system. International Journal of Control 18, 577–596 (1973)
25. Verghese, G.C., Kailath, T.: Rational matrix structure. IEEE Transactions on Automatic Control AC-26, 434–439 (1981)

Constructing Extended Formulations from Reflection Relations

Volker Kaibel and Kanstantsin Pashkovich*

Otto-von-Güricke-Universität Magdeburg, Institut für Mathematische Optimierung
Universitätsplatz 2, 39108 Magdeburg, Germany
{kaibel,pashkovi}@ovgu.de

Abstract. There are many examples of optimization problems whose associated polyhedra can be described much nicer, and with way less inequalities, by projections of higher dimensional polyhedra than this would be possible in the original space. However, currently not many general tools to construct such extended formulations are available. In this paper, we develop a framework of polyhedral relations that generalizes inductive constructions of extended formulations via projections, and we particularly elaborate on the special case of reflection relations. The latter ones provide polynomial size extended formulations for several polytopes that can be constructed as convex hulls of the unions of (exponentially) many copies of an input polytope obtained via sequences of reflections at hyperplanes. We demonstrate the use of the framework by deriving small extended formulations for the G-permutahedra of all finite reflection groups G (generalizing both Goeman's [6] extended formulation of the permutahedron of size $\mathrm{O}(n \log n)$ and Ben-Tal and Nemirovski's [2] extended formulation with $\mathrm{O}(k)$ inequalities for the regular 2^k-gon) and for Huffman-polytopes (the convex hulls of the weight-vectors of Huffman codes).

1 Introduction

An *extension* of a polyhedron $P \subseteq \mathbb{R}^n$ is some polyhedron $Q \subseteq \mathbb{R}^d$ and a linear projection $\pi : \mathbb{R}^d \to \mathbb{R}^n$ with $\pi(Q) = P$. A description of Q by linear inequalities (and equations) is called an *extended formulation* for P. Extended formulations have received quite some interest, as in several cases, one can describe polytopes associated with combinatorial optimization problems much easier by means of extended formulations than by linear descriptions in the original space. In particular, such extensions Q can have way less facets than the polyhedron P has. For a nice survey on extended formulations we refer to [4].

Many fundamental questions on the existence of extended formulations with small numbers of inequalities are open. A particularly prominent one asks whether there are polynomial size extended formulations for the perfect matching polytopes of complete graphs (see [14,9]). In fact, we lack good techniques to bound the sizes of extended formulations from below, and we also need more tools to construct extended formulations. This paper makes a contribution into the latter direction.

* Supported by the *International Max Planck Research School (IMPRS) for Analysis, Design and Optimization in Chemical and Biochemical Process Engineering Magdeburg*.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 287–300, 2011.

There are several ways to build extended formulations of polytopes from linear descriptions or from extended formulations of other ones (see, e.g., [10,8]). A particularly simple way is to construct them inductively from extended formulations one has already constructed before. As an example, let for a vector $p \in \mathbb{R}^n_+$ of *processing times* and for some $\sigma \in \mathfrak{S}(n)$ (where $\mathfrak{S}(n)$ is the set of all bijections $\gamma : [n] \to [n]$ with $[n] = \{1, \ldots, n\}$), the *completion time vector* be the vector $\mathrm{ct}(p, \sigma) \in \mathbb{R}^n$ with $\mathrm{ct}(p, \sigma)_j = \sum_{i=1}^{\sigma(j)} p_{\sigma^{-1}(i)}$ for all $j \in [n]$. Additionally, the *completion time polytope* $\mathrm{P}^p_{\mathrm{ct}}$ corresponding to the processing times vector $p \in \mathbb{R}^n_+$ is defined as

$$\mathrm{P}^p_{\mathrm{ct}} = \mathrm{conv}(\{\mathrm{ct}(p, \sigma) : \sigma \in \mathfrak{S}(n)\}) .$$

By some simple arguments, one can show that $\mathrm{P}^p_{\mathrm{ct}}$ is the image of the polytope $P = \mathrm{P}^{\tilde{p}}_{\mathrm{ct}} \times [0, 1]^{n-1}$ for $\tilde{p} = (p_1, \ldots, p_{n-1}) \in \mathbb{R}^{n-1}$ under the affine map $f : \mathbb{R}^{2n-2} \to \mathbb{R}^n$ defined via $f(x) = (x' + p_n x'', \langle \tilde{p}, \mathbf{1} - x'' \rangle + p_n)$ with $x = (x', x'')$ and $x', x'' \in \mathbb{R}^{n-1}$.

Applying this inductively, one finds that $\mathrm{P}^p_{\mathrm{ct}}$ is a *zonotope*, i.e., an affine projection of a cube of dimension $n(n-1)/2$ (which had already been proved by Wolsey in the 1980's [13]). This may appear surprisingly simple viewing the fact that $\mathrm{P}^p_{\mathrm{ct}}$ has exponentially many facets (see [12]). For the special case of the *permutahedron* $\mathrm{P}^n_{\mathrm{perm}} = \mathrm{P}^{\mathbf{1}_n}_{\mathrm{ct}} = \mathrm{conv}\{(\gamma(1), \ldots, \gamma(n)) \in \mathbb{R}^n : \gamma \in \mathfrak{S}(n)\}$, Goemans [6] found an even smaller extended formulation of size $\mathrm{O}(n \log n)$, which we will come back to later.

Let us look again at one step in the inductive construction described above. With the polyhedron

$$R = \{(x, y) \in \mathbb{R}^{2n-2} \times \mathbb{R}^n : y = f(x)\} , \tag{1}$$

the extension derived in such a step reads

$$\mathrm{P}^p_{\mathrm{ct}} = \{y \in \mathbb{R}^n : (x, y) \in R \text{ for some } x \in P\} . \tag{2}$$

Thus, we have derived the extended formulation for $\mathrm{P}^p_{\mathrm{ct}}$ by applying in the sense of (2) the "polyhedral relation" defined in (1) to a polytope P of which we had found (inductively) an extended formulation before. The goal of this paper is to generalize this technique of deriving extended formulations by using other "polyhedral relations" than graphs of affine maps (which R as defined in (1) is). We will introduce the framework of such general polyhedral relations in Section 2, and we are going to elaborate on one particular type of those, called *reflection relations*, in Section 3. Reflection relations provide, for affine halfspaces $H^{\leq} \subseteq \mathbb{R}^n$ and polyhedra $P \subseteq \mathbb{R}^n$, small extended formulations of the convex hull of the union of $P \cap H^{\leq}$ and the image of $P \cap H^{\leq}$ under the orthogonal reflection at the boundary hyperplane of $H^{\leq}$. They turn out to be quite useful building blocks in the construction of some extended formulations. We derive general results on reflection relations (Theorem 1) that allow to construct rather easily extended formulations for some particular applications (in particular, without explicitly dealing with the intermediate polyhedra of iterated constructions) .

In a first application, we show how to derive, for each polytope $P \subseteq \mathbb{R}^n$ that is contained in (the topological closure of) a fundamental region of a finite reflection group G on $\mathbb{R}^n$, an extended formulation of the G-permutahedron of P, i.e., the convex hull of the union of the polytopes in the orbit of P under the action of G (Section 4.1). These

extended formulations have $f' + \mathrm{O}(n \log n) + \mathrm{O}(n \log m)$ inequalities, where m is the largest number such that $I_2(m)$ appears in the decomposition of G into irreducible finite reflection groups, and provided that there is an extended formulation for P with at most f' inequalities. In particular, this generalizes Goemans' extended formulation of the permutahedron $\mathrm{P}^n_{\mathrm{perm}}$ with $\mathrm{O}(n \log n)$ inequalities [6]. In fact, the starting point of our research was to give an alternative proof for the correctness of Goeman's extended formulation that we would be able to generalize to other constructions.

As a second application, we provide an extended formulation with $\mathrm{O}(n \log n)$ inequalities for the convex hull of all weight-vectors of Huffman-codes with n words (Section 4.2). This *Huffman-polytope* $\mathrm{P}^n_{\mathrm{huff}} \subseteq \mathbb{R}^n$ is the convex hull of all vectors $(v_1, \ldots, v_n)$ for which there is a rooted binary tree with n leaves labelled in arbitrary order by $1, \ldots, n$ such that the distance of leaf i from the root equals v_i for all $i \in [n]$. This provides another striking example of the power of extended formulations, as no linear descriptions of $\mathrm{P}^n_{\mathrm{huff}}$ in $\mathbb{R}^n$ is known so far, and Nguyen, Nguyen, and Maurras [11] showed that $\mathrm{P}^n_{\mathrm{huff}}$ has $2^{\Omega(n \log n)}$ facets.

Two well-known results we obtain easily within the framework of reflection relations are extended formulations with $2\lceil \log(m) \rceil + 2$ inequalities for regular m-gons (reproving a result of Ben-Tal and Nemirovski [2], see Section 4.1) and an extended formulation with $4(n-1)$ inequalities of the *parity polytope*, i.e., the convex hull of all $v \in \{0,1\}^n$ with an odd number of one-entries (reproving a result of Carr and Konjevod [3], see Section 4.1).

We conclude by briefly discussing (Section 5) directions for future research on the further extension of the tools presented in this paper .

All extended formulations described in this paper can also be constructed efficiently, i.e., in a number of steps that is bounded by a polynomial in the size of the formulation (assuming symbolic encoding of $\sin(\varphi)$ and $\cos(\varphi)$ in the formulations of the G-permutahedra for $G = I_2(m)$).

2 Polyhedral Relations

A *polyhedral relation* of *type* (n, m) is a non-empty polyhedron $\varnothing \neq R \subseteq \mathbb{R}^n \times \mathbb{R}^m$. The *image* of a subset $X \subseteq \mathbb{R}^n$ under such a polyhedral relation R is denoted by

$$R(X) = \{y \in \mathbb{R}^m : (x, y) \in R \text{ for some } x \in X\}.$$

Clearly, we have the monotonicity relations $R(X) \subseteq R(\tilde{X})$ for $X \subseteq \tilde{X}$. Furthermore, $R(X)$ is a linear projection of $R \cap (X \times \mathbb{R}^m)$. Thus, images of polyhedra and convex sets under polyhedral relations are polyhedra and convex sets, respectively.

A *sequential polyhedral relation* of *type* $(k_0, \ldots, k_r)$ is a sequence $R_1, \ldots, R_r$, where R_i is a polyhedral relation of type (k_{i-1}, k_i) for each $i \in [r]$; its *length* is r. For such a sequential polyhedral relation, we denote by $\mathcal{R} = R_r \circ \cdots \circ R_1$ the set of all $(z^{(0)}, z^{(r)}) \in \mathbb{R}^{k_0} \times \mathbb{R}^{k_r}$ for which there is some $(z^{(1)}, \ldots, z^{(r-1)})$ with $(z^{(i-1)}, z^{(i)}) \in R_i$ for all $i \in [r]$. Since $\mathcal{R}$ is a linear projection of a polyhedron it is a polyhedral relation of type (k_0, k_r) with $R_r \circ \cdots \circ R_1(X) = R_r(\cdots R_1(X) \cdots)$ for all $X \subseteq \mathbb{R}^{k_0}$.

We call $\mathcal{R} = R_r \circ \cdots \circ R_1$ the polyhedral relation that is *induced* by the sequential polyhedral relation $R_1, \ldots, R_r$. For a polyhedron $P \subseteq \mathbb{R}^{k_0}$, the polyhedron $Q \subseteq \mathbb{R}^{k_0} \times \cdots \times \mathbb{R}^{k_r}$ defined by

$$z^{(0)} \in P \quad \text{and} \quad (z^{(i-1)}, z^{(i)}) \in R_i \quad \text{for all } i \in [r] \tag{3}$$

satisfies $\pi(Q) = \mathcal{R}(P)$, where π is the projection defined via $\pi(z^{(0)}, \ldots, z^{(r)}) = z^{(r)}$. Thus, (3) provides an extended formulation of the polyhedron $\mathcal{R}(P)$ with $k_0 + \cdots + k_r$ variables and $f_0 + \cdots + f_r$ constraints, provided we have linear descriptions of the polyhedra P, R_1, …, R_r with f_0, f_1, …, f_r constraints, respectively. Of course, one can reduce the number of variables in this extended formulation to $\dim(Q)$. In order to obtain useful upper bounds on this number by means of the polyhedral relations R_1, …, R_r, let us denote, for any polyhedral relation $R \subseteq \mathbb{R}^n \times \mathbb{R}^m$, by $\delta_1(R)$ and $\delta_2(R)$ the dimension of the non-empty fibers of the orthogonal projection of $\mathrm{aff}(R)$ to the first and second factor of $\mathbb{R}^n \times \mathbb{R}^m$, respectively. If $\mathrm{aff}(R) = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m : Ax + By = c\}$, then $\delta_1(R) = \dim(\ker(B))$ and $\delta_2(R) = \dim(\ker(A))$. With these parameters, we can estimate

$$\dim(Q) \leq \min\{k_0 + \sum_{i=1}^{r} \delta_1(R_i), k_r + \sum_{i=1}^{r} \delta_2(R_i)\}\,.$$

Remark 1. *If $R_1, \ldots, R_r$ is a sequential polyhedral relation of type $(k_0, \ldots, k_r)$ with induced polyhedral relation $\mathcal{R} = R_r \circ \cdots \circ R_1$, let $\pi : \mathbb{R}^{k_0} \times \cdots \times \mathbb{R}^{k_r} \to \mathbb{R}^{k_r}$ be the projection defined via $\pi(z^{(0)}, \ldots, z^{(r)}) = z^{(r)}$, and let f_i be the number of facets of R_i for each $i \in [r]$. If the polyhedron $P \subseteq \mathbb{R}^{k_0}$ has an extended formulation with k' variables and f' inequalities, then we can construct an extended formulation for $\mathcal{R}(P)$ with $\min\{k' + \sum_{i=1}^{r} \delta_1(R_i), k_r + \sum_{i=1}^{r} \delta_2(R_i)\}$ variables and $f' + f_1 + \cdots + f_r$ constraints.*

A particularly simple class of polyhedral relations is defined by polyhedra $R \subseteq \mathbb{R}^n \times \mathbb{R}^m$ with $R = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m : y = f(x)\}$ for some affine map $f : \mathbb{R}^n \to \mathbb{R}^m$. For these polyhedral relations, a (linear description of a) polyhedron $P \subseteq \mathbb{R}^n$ is just an extended formulation of the polyhedron $R(P)$ via projection f.

The *domain* of a polyhedral relation $R \subseteq \mathbb{R}^n \times \mathbb{R}^m$ is the polyhedron

$$\mathrm{dom}(R) = \{x \in \mathbb{R}^n : (x, y) \in R \text{ for some } y \in \mathbb{R}^m\}\,.$$

We clearly have $R(X) = \bigcup_{x \in X \cap \mathrm{dom}(R)} R(x)$ for all $X \subseteq \mathbb{R}^n$. Note that, for a polytope $P = \mathrm{conv}(V)$ with a finite set $V \subseteq \mathbb{R}^n$ and a polyhedral relation $R \subseteq \mathbb{R}^n \times \mathbb{R}^m$, in general the inclusion

$$\mathrm{conv} \bigcup_{v \in V} R(v) \subseteq R(P) \tag{4}$$

holds without equality, even in case of $P \subseteq \mathrm{dom}(R)$; as for an example you may consider $P = \mathrm{conv}\{0, 2\} \subseteq \mathbb{R}^1$ and $R = \mathrm{conv}\{(0, 0), (1, 1), (2, 0)\}$ with $R(P) = [0, 1]$ and $R(0) = R(2) = \{0\}$. Fortunately, one can guarantee equality in (4) (which makes it much easier to analyze $R(P)$) for an important subclass of polyhedral relations.

We call a relation $R \subseteq \mathbb{R}^n \times \mathbb{R}^m$ *affinely generated* by the family $(\varrho^{(f)})_{f\in F}$, if F is finite and every $\varrho^{(f)} : \mathbb{R}^n \to \mathbb{R}^m$ is an affine map such that $R(x) = \operatorname{conv} \bigcup_{f\in F} \varrho^{(f)}(x)$ holds for all $x \in \operatorname{dom}(R)$. The maps $\varrho^{(f)}$ ($f \in F$) are called *affine generators* of R in this case. For such a polyhedral relation R and a polytope $P \subseteq \mathbb{R}^n$ with $P \cap \operatorname{dom}(R) = \operatorname{conv}(V)$ for some $V \subseteq \mathbb{R}^n$, we find

$$\begin{aligned} R(P) &= \bigcup_{x\in P\cap\operatorname{dom}(R)} R(x) = \bigcup_{x\in P\cap\operatorname{dom}(R)} \operatorname{conv} \bigcup_{f\in F} \varrho^{(f)}(x) \\ &\subseteq \operatorname{conv} \bigcup_{x\in P\cap\operatorname{dom}(R)} \bigcup_{f\in F} \varrho^{(f)}(x) = \operatorname{conv} \bigcup_{v\in V} \bigcup_{f\in F} \varrho^{(f)}(v) \subseteq \operatorname{conv} \bigcup_{v\in V} R(v)\,, \end{aligned}$$

where, due to (4), all inclusions are equations. In particular, we have established the following result.

Proposition 1. *For every polyhedral relation $R \subseteq \mathbb{R}^n \times \mathbb{R}^m$ that is affinely generated by a finite family $(\varrho^{(f)})_{f\in F}$, and for every polytope $P \subseteq \mathbb{R}^n$, we have*

$$R(P) = \operatorname{conv} \bigcup_{f\in F} \varrho^{(f)}(P \cap \operatorname{dom}(R))\,. \tag{5}$$

As we will often deal with polyhedral relations $\mathcal{R} = R_r \circ \cdots \circ R_1$ that are induced by a sequential polyhedral relation $R_1, \ldots, R_r$, it would be convenient to be able to derive affine generators for $\mathcal{R}$ from affine generators for $R_1,\ldots,R_r$. This, however, seems impossible in general, where the difficulties arise from the interplay between images and domains in a sequence of polyhedral relations. However, one still can derive a very useful analogue of the inclusion "$\subseteq$" in (5).

Lemma 1. *If we have $\mathcal{R} = R_r \circ \cdots \circ R_1$ and for each $i \in [r]$ the relation R_i is affinely generated by the finite family $(\varrho^{(f_i)})_{f_i\in F_i}$, then the inclusion*

$$\mathcal{R}(P) \subseteq \operatorname{conv} \bigcup_{f\in F} \varrho^{(f)}(P \cap \operatorname{dom}(\mathcal{R}))$$

holds for every polyhedron $P \subseteq \mathbb{R}^n$, where $F = F_1 \times \cdots \times F_r$ and $\varrho^{(f)} = \varrho^{(f_r)} \circ \cdots \circ \varrho^{(f_1)}$ for each $f = (f_1, \ldots, f_r) \in F$.

We omit the straight-forward proof of Lemma 1 in this extended abstract.

3 Reflection Relations

For $a \in \mathbb{R}^n \setminus \{\mathbf{O}\}$ and $\beta \in \mathbb{R}$, we denote by $\mathrm{H}^{=}(a, \beta) = \{x \in \mathbb{R}^n : \langle a, x\rangle = \beta\}$ the hyperplane defined by the equation $\langle a, x\rangle = \beta$ and by $\mathrm{H}^{\leq}(a, \beta) = \{x \in \mathbb{R}^n : \langle a, x\rangle \leq \beta\}$ the halfspace defined by the inequality $\langle a, x\rangle \leq \beta$ (with $\langle v, w\rangle = \sum_{i=1}^n v_i w_i$ for all $v, w \in \mathbb{R}^n$). The reflection at $H = \mathrm{H}^{=}(a, \beta)$ is $\varrho^{(H)} : \mathbb{R}^n \to \mathbb{R}^n$ where $\varrho^{(H)}(x)$

is the point with $\varrho^{(H)}(x) - x \in H^\perp$ lying in the one-dimensional linear subspace $H^\perp = \{\lambda a : \lambda \in \mathbb{R}\}$ that is orthogonal to H and $\langle a, \varrho^{(H)}(x)\rangle = 2\beta - \langle a, x\rangle$. The *reflection relation* defined by (a, β) is

$$\mathrm{R}_{a,\beta} = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^n : y - x \in (\mathrm{H}^=(a, \beta))^\perp, \langle a, x\rangle \le \langle a, y\rangle \le 2\beta - \langle a, x\rangle\}$$

(the definition is invariant against scaling (a, β) by positive scalars). For the halfspace $H^\le = \mathrm{H}^\le(a, \beta)$, we also denote $\mathrm{R}_{H^\le} = \mathrm{R}_{a,\beta}$. The domain of the reflection relation is $\mathrm{dom}(\mathrm{R}_{a,\beta}) = H^\le$, as $(x, y) \in \mathrm{R}_{a,\beta}$ implies $\langle a, x\rangle \le 2\beta - \langle a, x\rangle$, thus $\langle a, x\rangle \le \beta$, and furthermore, for each $x \in \mathrm{H}^\le(a, \beta)$, we obviously have $(x, x) \in \mathrm{R}_{a,\beta}$. Note that, although (a, β) and $(-a, -\beta)$ define the same reflection, the reflection relations $\mathrm{R}_{a,\beta}$ and $\mathrm{R}_{-a,-\beta}$ have different domains.

From the constraint $y - x \in (\mathrm{H}^=(a, \beta))^\perp$ it follows that $\delta_1(\mathrm{R}_{a,\beta}) = 1$ holds. Thus, we can deduce the following from Remark 1.

Remark 2. *If $\mathcal{R}$ is induced by a sequential polyhedral relation of type $(n, \ldots, n)$ and length r consisting of reflection relations only, then, for every polyhedron $P \subseteq \mathbb{R}^n$, an extended formulation of $\mathcal{R}(P)$ with $n' + r$ variables and $f' + 2r$ inequalities can be constructed, provided one has at hands an extended formulation for P with n' variables and f' inequalities.*

Proposition 2. *For $a \in \mathbb{R}^n \setminus \{\mathbf{O}\}$, $\beta \in \mathbb{R}$ and the hyperplane $H = \mathrm{H}^=(a, \beta)$, the reflection relation $\mathrm{R}_{a,\beta}$ is affinely generated by the identity map and the reflection $\varrho^{(H)}$.*

Proof. We need to show $\mathrm{R}_{a,\beta}(x) = \mathrm{conv}\{x, \varrho^{(H)}(x)\}$ for every $x \in \mathrm{dom}(\mathrm{R}_{a,\beta}) = \mathrm{H}^\le(a, \beta)$. Since, for each such x, we have $(x, x) \in \mathrm{R}_{a,\beta}(x)$ and $(x, \varrho^{(H)}(x)) \in \mathrm{R}_{a,\beta}(x)$, and due to the convexity of $\mathrm{R}_{a,\beta}(x)$, it suffices to establish the inclusion "$\subseteq$". Thus, let $y \in \mathrm{R}_{a,\beta}(x)$ be an arbitrary point in $\mathrm{R}_{a,\beta}(x)$. Due to $\varrho^{(H)}(x) - x \in H^\perp$ and $y - x \in H^\perp$, both x and $\varrho^{(H)}(x)$ are contained in the line $y + H^\perp$. From $2\beta - \langle a, x\rangle = \langle a, \varrho^{(H)}(x)\rangle$ and $\langle a, x\rangle \le \langle a, y\rangle \le 2\beta - \langle a, x\rangle$ we hence conclude that y is a convex combination of x and $\varrho^{(H)}(x)$.

From Proposition 1 and Proposition 2, one obtains the following result.

Corollary 1. *If $P \subseteq \mathbb{R}^n$ is a polytope, then we have, for $a \in \mathbb{R}^n \setminus \{\mathbf{O}\}$ and $\beta \in \mathbb{R}$ defining the hyperplane $H = \mathrm{H}^=(a, \beta)$ and the halfspace $H^\le = \mathrm{H}^\le(a, \beta)$,*

$$\mathrm{R}_{a,\beta}(P) = \mathrm{conv}\left((P \cap H^\le) \cup \varrho^{(H)}(P \cap H^\le)\right).$$

While Corollary 1 describes images under single reflection relations, for analyses of the images under sequences of reflection relations we define, for each $a \in \mathbb{R}^n \setminus \{\mathbf{O}\}$, $\beta \in \mathbb{R}$, $H^\le = \mathrm{H}^\le(a, \beta)$, and $H = \mathrm{H}^=(a, \beta)$, the map $\varrho^{\star(H^\le)} : \mathbb{R}^n \to \mathbb{R}^n$ via

$$\varrho^{\star(H^\le)}(y) = \begin{cases} y & \text{if } y \in H^\le \\ \varrho^{(H)}(y) & \text{otherwise} \end{cases}$$

for all $y \in \mathbb{R}^n$, which assigns a canonical preimage to every $y \in \mathbb{R}^n$. If $\mathcal{R}$ denotes the polyhedral relation $\mathrm{R}_{H_r^\le} \circ \ldots \circ \mathrm{R}_{H_1^\le}$, then we have

$$y \in \mathcal{R}(\varrho^{\star(H_1^\le)} \circ \cdots \circ \varrho^{\star(H_r^\le)}(y)) \tag{6}$$

for all $y \in \mathbb{R}^n$.

Theorem 1. *For* $\mathcal{R} = \mathrm{R}_{H_r^{\leq}} \circ \ldots \circ \mathrm{R}_{H_1^{\leq}}$ *with halfspaces* $H_1^{\leq}, \ldots, H_r^{\leq} \subseteq \mathbb{R}^n$ *and boundary hyperplanes* $H_1, \ldots, H_r$ *as well as polytopes* $P, Q \subseteq \mathbb{R}^n$ *with* $Q = \mathrm{conv}(W)$ *for some* $W \subseteq \mathbb{R}^n$ *we have* $Q = \mathcal{R}(P)$ *whenever the following two conditions are satisfied:*

1. *We have* $P \subseteq Q$ *and* $\varrho^{(H_i)}(Q) \subseteq Q$ *for all* $i \in [r]$.
2. *We have* $\varrho^{\star(H_1^{\leq})} \circ \cdots \circ \varrho^{\star(H_r^{\leq})}(w) \in P$ *for all* $w \in W$.

Proof. From the first condition it follows that the image of P under every combination of maps $\varrho^{(H_i)}$ lies in Q. Thus, from Lemma 1 we have the inclusion $\mathcal{R}(P) \subseteq Q$. By the second condition and (6), we have $W \subseteq \mathcal{R}(P)$, and hence $Q = \mathrm{conv}(W) \subseteq \mathcal{R}(P)$ due to the convexity of $\mathcal{R}(P)$.

In order to provide simple examples of extended formulations obtained from reflection relations, let us define the *signing* of a polyhedron $P \subseteq \mathbb{R}^n$ to be

$$\mathrm{sign}(P) = \mathrm{conv} \bigcup_{\epsilon \in \{-,+\}^n} \epsilon.P\,,$$

where $\epsilon.x$ is the vector obtained from $x \in \mathbb{R}^n$ by changing the signs of all coordinates i with ϵ_i being minus. For $x \in \mathbb{R}^n$, we denote by $x^{(\mathrm{abs})} \in \mathbb{R}^n$ the vector that is obtained from x by changing every component to its absolute value.

For the construction below we use the reflection relations $\mathrm{R}_{-\mathbf{e}_k,0}$ (where $\mathbf{e}_k$ is the k-th standard unit vector), denoted by S_k, for all $k \in [n]$. The corresponding reflection $\sigma_k : \mathbb{R}^n \to \mathbb{R}^n$ is just the sign change of the k-th coordinate, given by

$$\sigma_k(x)_i = \begin{cases} -x_i & \text{if } i = k \\ x_i & \text{otherwise} \end{cases}$$

for all $x \in \mathbb{R}^n$. The map which defines the canonical preimage with respect to the relation S_k is given by

$$\sigma_k^{\star}(y)_i = \begin{cases} |y_i| & \text{if } i = k \\ y_i & \text{otherwise} \end{cases}$$

for all $y \in \mathbb{R}^n$.

Proposition 3. *If* $\mathcal{R}$ *is the polyhedral relation* $\mathrm{S}_n \circ \ldots \circ \mathrm{S}_1$ *and* $P \subseteq \mathbb{R}^n$ *is a polytope with* $v^{(abs)} \in P$ *for each vertex* v *of* P*, then we have*

$$\mathcal{R}(P) = \mathrm{sign}(P)\,.$$

Proof. With $Q = \mathrm{sign}(P)$, the first condition of Theorem 1 is satisfied. Furthermore, we have $Q = \mathrm{conv}(W)$ with $W = \{\epsilon.v : \epsilon \in \{-,+\}^n, v \text{ vertex of } P\}$. As, for every $w \in W$ with $w = \epsilon.v$ for some vertex v of P and $\epsilon \in \{-,+\}^n$, we have $\sigma_1^{\star} \circ \cdots \circ \sigma_n^{\star}(w) = w^{(\mathrm{abs})} = v^{(\mathrm{abs})} \in P$, also the second condition of Theorem 1 is satisfied. Hence the claim follows.

Proposition 3 and Remark 2 imply the following.

Theorem 2. *For each polytope* $P \subseteq \mathbb{R}^n$ *with* $v^{(abs)} \in P$ *for each vertex* v *of* P *there is an extended formulation of* $\mathrm{sign}(P)$ *with* $n' + n$ *variables and* $f' + 2n$ *inequalities, whenever* P *admits an extended formulation with* n' *variables and* f' *inequalities.*

4 Applications

4.1 Reflection Groups

A *finite reflection group* is a group G of finite cardinality that is generated by a (finite) family $\varrho^{(H_i)} : \mathbb{R}^n \to \mathbb{R}^n$ $(i \in I)$ of reflections at hyperplanes $\mathbf{O} \in H_i \subseteq \mathbb{R}^n$ containing the origin. We refer to [7,5] for all results on reflection groups that we will mention. The set of *reflection hyperplanes* $H \subseteq \mathbb{R}^n$ with $\varrho^{(H)} \in G$ (and thus $\mathbf{O} \in H$) — called the *Coxeter arrangement* of G — cuts $\mathbb{R}^n$ into open connected components, which are called the *regions* of G. The group G is in bijection with the set of its regions, and it acts transitively on these regions. If one distinguishes arbitrarily the topological closure of one of them as the *fundamental domain* Φ_G of G, then, for every point $x \in \mathbb{R}^n$, there is a *unique* point $x^{(\Phi_G)} \in \Phi_G$ that belongs to the orbit of x under the action of the group G on $\mathbb{R}^n$.

A finite reflection group G is called *irreducible* if the set of reflection hyperplanes cannot be partitioned into two sets $\mathcal{H}_1$ and $\mathcal{H}_2$ such that the normal vectors of all hyperplanes in $\mathcal{H}_1$ are orthogonal to the normal vectors of all hyperplanes from $\mathcal{H}_2$. According to a central classification result, up to linear transformations, the family of irreducible finite reflection groups consists of the four infinite subfamilies $I_2(m)$ (on $\mathbb{R}^2$), A_{n-1}, B_n, and D_n (on $\mathbb{R}^n$), as well as six special groups.

For a finite reflection group G on $\mathbb{R}^n$ and some polytope $P \subseteq \mathbb{R}^n$ of G, the *G-permutahedron* $\Pi_G(P)$ of P is the convex hull of the union of the orbit of P under the action of G. In this subsection, we show for G being one of $I_2(m)$, A_{n-1}, B_n, or D_n, how to construct an extended formulation for $\Pi_G(P)$ from an extended formulation for P. The numbers of inequalities in the constructed extended formulations will be bounded by $f' + \mathrm{O}(\log m)$ in case of $G = I_2(m)$ and by $f' + \mathrm{O}(n \log n)$ in the other cases, provided that we have at hands an extended formulation of P with f' inequalities. By the decomposition into irreducible finite reflection groups, one can extend these constructions to arbitrary finite reflection groups G on $\mathbb{R}^n$, where the resulting extended formulations have $f' + \mathrm{O}(n \log m) + \mathrm{O}(n \log n)$ inequalities, where m is the largest number such that $I_2(m)$ appears in the decomposition of G into irreducible finite reflection groups. Details on this will be in the full version of the paper.

The Reflection Group $I_2(m)$. For $\varphi \in \mathbb{R}$, let us denote $H_\varphi = \mathrm{H}^{=}((-\sin\varphi, \cos\varphi), 0)$ and $H_\varphi^{\leq} = \mathrm{H}^{\leq}((-\sin\varphi, \cos\varphi), 0)$. The group $I_2(m)$ is generated by the reflections at H_0 and $H_{\pi/m}$. It is the symmetry group of the regular m-gon with its center at the origin and one of its vertices at $(1, 0)$. The group $I_2(m)$ consists of the (finite) set of all reflections $\varrho^{(H_{k\pi/m})}$ (for $k \in \mathbb{Z}$) and the (finite) set of all rotations around the origin by angles $2k\pi/m$ (for $k \in \mathbb{Z}$). We choose $\Phi_{I_2(m)} = \{x \in \mathbb{R}^2 : x_2 \geq 0, x \in H_{\pi/m}^{\leq}\}$ as the fundamental domain.

Proposition 4. *If $\mathcal{R}$ is the polyhedral relation* $\mathrm{R}_{H_{2^r\pi/m}^{\leq}} \circ \cdots \circ \mathrm{R}_{H_{2\pi/m}^{\leq}} \circ \mathrm{R}_{H_{\pi/m}^{\leq}}$ *with $r = \lceil \log(m) \rceil$ and $P \subseteq \mathbb{R}^2$ is a polytope with $v^{(\Phi_{I_2(m)})} \in P$ for each vertex v of P, then we have $\mathcal{R}(P) = \Pi_{I_2(m)}(P)$.*

Proof. With $Q = \Pi_{I_2(m)}(P)$, the first condition of Theorem 1 is satisfied. Furthermore, we have $Q = \operatorname{conv}(W)$ with $W = \{\gamma.v : \gamma \in I_2(m), v \text{ vertex of } P\}$. Let $w \in W$ be some point with $w = \gamma.v$ for some vertex v of P and $\gamma \in I_2(m)$. Observing that

$$\varrho^{\star(H^{\leq}_{\pi/m})} \circ \varrho^{\star(H^{\leq}_{2\pi/m})} \circ \cdots \circ \varrho^{\star(H^{\leq}_{2^r\pi/m})}(w)$$

is contained in $\Phi_{I_2(m)}$, we conclude that it equals $w^{(\Phi_{I_2(m)})} = v^{(\Phi_{I_2(m)})} \in P$. Therefore, also the second condition of Theorem 1 is satisfied. Hence the claim follows.

From Proposition 4 and Remark 2, we can conclude the following theorem.

Theorem 3. *For each polytope $P \subseteq \mathbb{R}^2$ with $v^{(\Phi_{I_2(m)})} \in P$ for each vertex v of P there is an extended formulation of $\Pi_{I_2(m)}(P)$ with $n' + \lceil \log(m) \rceil + 1$ variables and $f' + 2\lceil \log(m) \rceil + 2$ inequalities, whenever P admits an extended formulation with n' variables and f' inequalities.*

In particular, we obtain an extended formulation of a regular m-gon with $\lceil \log(m) \rceil + 1$ variables and $2\lceil \log(m) \rceil + 2$ inequalities by choosing $P = \{(1, 0)\}$ in Theorem 3, thus reproving a result due to Ben-Tal and Nemirovski [2].

The Reflection Group A_{n-1}. The group A_{n-1} is generated by the reflections in $\mathbb{R}^n$ at the hyperplanes $\mathrm{H}^{=}(\mathbf{e}_k - \mathbf{e}_\ell, 0)$ for all pairwise distinct $k, \ell \in [n]$. It is the symmetry group of the $(n-1)$-dimensional (hence the index in the notation A_{n-1}) simplex $\operatorname{conv}\{\mathbf{e}_1, \ldots, \mathbf{e}_n\} \subseteq \mathbb{R}^n$. We choose $\Phi_{A_{n-1}} = \{x \in \mathbb{R}^n : x_1 \leq \cdots \leq x_n\}$ as the fundamental domain. The orbit of a point $x \in \mathbb{R}^n$ under the action of A_{n-1} consists of all points which can be obtained from x by permuting coordinates. Thus the A_{n-1}-permutahedron of a polytope $P \subseteq \mathbb{R}^n$ is

$$\Pi_{A_{n-1}}(P) = \operatorname{conv} \bigcup_{\gamma \in \mathfrak{S}(n)} \gamma.P\,,$$

where $\gamma.x$ is the vector obtained from $x \in \mathbb{R}^n$ by permuting the coordinates according to γ.

Let us consider more closely the reflection relation $\mathrm{T}_{k,\ell} = \mathrm{R}_{\mathbf{e}_k - \mathbf{e}_\ell, 0} \subseteq \mathbb{R}^n \times \mathbb{R}^n$. The corresponding reflection $\tau_{k,\ell} = \varrho^{(H_{k,\ell})} : \mathbb{R}^n \to \mathbb{R}^n$ with $H_{k,\ell} = \mathrm{H}^{=}(\mathbf{e}_k - \mathbf{e}_\ell, 0)$ is the transposition of coordinates k and ℓ, i.e., we have

$$\tau_{k,\ell}(x)_i = \begin{cases} x_\ell & \text{if } i = k \\ x_k & \text{if } i = \ell \\ x_i & \text{otherwise} \end{cases}$$

for all $x \in \mathbb{R}^n$. The map $\tau^\star_{k,\ell} = \varrho^{\star(H_{k,\ell})} : \mathbb{R}^n \to \mathbb{R}^n$ (assigning canonical preimages) is given by

$$\tau^\star_{k,\ell}(y) = \begin{cases} \tau_{k,\ell}(y) & \text{if } y_k > y_\ell \\ y & \text{otherwise} \end{cases}$$

for all $y \in \mathbb{R}^n$.

A sequence $(k_1, \ell_1), \ldots, (k_r, \ell_r) \in [n] \times [n]$ with $k_i \neq \ell_i$ for all $i \in [r]$ is called a *sorting network* if $\tau^\star_{k_1,\ell_1} \circ \cdots \circ \tau^\star_{k_r,\ell_r}(y) = y^{(\text{sort})}$ holds for all $y \in \mathbb{R}^n$, where we denote by $y^{(\text{sort})} \in \mathbb{R}^n$ the vector that is obtained from y by sorting the components in non-decreasing order. Note that we have $y^{(\Phi_{A_{n-1}})} = y^{(\text{sort})}$ for all $y \in \mathbb{R}^n$.

Proposition 5. *If $\mathcal{R}$ is a polyhedral relation $\mathrm{T}_{k_r,\ell_r} \circ \ldots \circ \mathrm{T}_{k_1,\ell_1}$, where the sequence $(k_1, \ell_1), \ldots, (k_r, \ell_r) \in [n] \times [n]$ is a sorting network, and $P \subseteq \mathbb{R}^n$ is a polytope with $v^{(sort)} \in P$ for each vertex v of P, then we have $\mathcal{R}(P) = \Pi_{A_{n-1}}(P)$.*

Proof. With $Q = \Pi_{A_{n-1}}(P)$, the first condition of Theorem 1 is satisfied. Furthermore, we have $Q = \mathrm{conv}(W)$ with $W = \{\gamma.v : \gamma \in \mathfrak{S}(n), v \text{ vertex of } P\}$. As, for every $w \in W$ with $w = \gamma.v$ for some vertex v of P and $\gamma \in \mathfrak{S}(n)$, we have

$$\tau^\star_{k_1,\ell_1} \circ \cdots \circ \tau^\star_{k_r,\ell_r}(w) = w^{(\text{sort})} = v^{(\text{sort})} \in P\,,$$

also the second condition of Theorem 1 is satisfied. Hence the claim follows.

As there are sorting networks of size $r = \mathrm{O}(n \log n)$ (see [1]), from Proposition 5 and Remark 2 we can conclude the following theorem.

Theorem 4. *For each polytope $P \subseteq \mathbb{R}^n$ with $v^{(sort)} \in P$ for each vertex v of P there is an extended formulation of $\Pi_{A_{n-1}}(P)$ with $n' + \mathrm{O}(n \log n)$ variables and $f' + \mathrm{O}(n \log n)$ inequalities, whenever P admits an extended formulation with n' variables and f' inequalities.*

Note that the sorting networks described in [1] can be computed in time that is bounded polynomially in n.

Choosing the one-point polytope $P = \{(1, 2, \ldots, n)\} \subseteq \mathbb{R}^n$, Theorem 4 yields basically the same extended formulation with $\mathrm{O}(n \log n)$ variables and inequalities of the permutahedron $\mathrm{P}^n_{\text{perm}} = \Pi_{A_{n-1}}(P)$ that has been constructed by Goemans [6] (see the remarks in the Introduction).

The Reflection Group B_n. The group B_n is generated by the reflections in $\mathbb{R}^n$ at the hyperplanes $\mathrm{H}^=(\mathbf{e}_k + \mathbf{e}_\ell, 0)$, $\mathrm{H}^=(\mathbf{e}_k - \mathbf{e}_\ell, 0)$ and $\mathrm{H}^=(\mathbf{e}_k, 0)$ for all pairwise distinct $k, \ell \in [n]$. It is the symmetry group of both the n-dimensional cube $\mathrm{conv}\{-1, +1\}^n$ and the n-dimensional cross-polytope $\mathrm{conv}\{\pm\mathbf{e}_1, \ldots, \pm\mathbf{e}_n\}$. We choose $\Phi_{B_n} = \{x \in \mathbb{R}^n : 0 \leq x_1 \leq \cdots \leq x_n\}$ as the fundamental domain. The orbit of a point $x \in \mathbb{R}^n$ under the action of B_n consists of all points which can be obtained from x by permuting its coordinates and changing the signs of some subset of its coordinates. Note that we have $y^{(\Phi_{B_n})} = y^{(\text{sort-abs})}$ for all $y \in \mathbb{R}^n$, where $y^{(\text{sort-abs})} = v'^{(\text{sort})}$ with $v' = v^{(\text{abs})}$.

Proposition 6. *If $\mathcal{R}$ is a polyhedral relation $S_n \circ \ldots \circ S_1 \circ \mathrm{T}_{k_r,\ell_r} \circ \ldots \circ \mathrm{T}_{k_1,\ell_1}$, where $(k_1, \ell_1), \ldots, (k_r, \ell_r) \in [n] \times [n]$ is a sorting network (and the S_i are defined as at the end of Section 3) and $P \subseteq \mathbb{R}^n$ is a polytope with $v^{(sort\text{-}abs)} \in P$ for each vertex v of P, then we have $\mathcal{R}(P) = \Pi_{B_n}(P)$.*

Proof. With $Q = \Pi_{B_n}(P)$, the first condition of Theorem 1 is satisfied. Furthermore, we have $Q = \operatorname{conv}(W)$ with $W = \{\gamma.\epsilon.v : \gamma \in \mathfrak{S}(n), \epsilon \in \{-,+\}^n, v \text{ vertex of } P\}$. As, for every $w \in W$ with $w = \gamma.\epsilon.v$ for some vertex v of P and $\gamma \in \mathfrak{S}(n)$, $\epsilon \in \{-,+\}^n$, we have

$$\tau^\star_{k_1,\ell_1} \circ \cdots \circ \tau^\star_{k_r,\ell_r} \circ \sigma^\star_1 \circ \cdots \circ \sigma^\star_n(w) = w^{(\text{sort-abs})} = v^{(\text{sort-abs})} \in P\,,$$

also the second condition of Theorem 1 is satisfied. Hence the claim follows.

As for A_{n-1}, we thus can conclude the following from Proposition 6 and Remark 2.

Theorem 5. *For each polytope $P \subseteq \mathbb{R}^n$ with $v^{(\text{sort-abs})} \in P$ for each vertex v of P there is an extended formulation of $\Pi_{B_n}(P)$ with $n' + \mathrm{O}(n \log n)$ variables and $f' + \mathrm{O}(n \log n)$ inequalities, whenever P admits an extended formulation with n' variables and f' inequalities.*

The Reflection Group D_n. The group D_n is generated by the reflections in $\mathbb{R}^n$ at the hyperplanes $\mathrm{H}^=(\mathbf{e}_k + \mathbf{e}_\ell, 0)$ and $\mathrm{H}^=(\mathbf{e}_k - \mathbf{e}_\ell, 0)$ for all pairwise distinct $k, \ell \in [n]$. Thus, D_n is a proper subgroup of B_n. It is not the symmetry group of a polytope. We choose $\Phi_{D_n} = \{x \in \mathbb{R}^n : |x_1| \le x_2 \le \cdots \le x_n\}$ as the fundamental domain. The orbit of a point $x \in \mathbb{R}^n$ under the action of D_n consists of all points which can be obtained from x by permuting its coordinates and changing the signs of an *even* number of its coordinates. For every $x \in \mathbb{R}^n$, the point $x^{(\Phi_{D_n})}$ arises from $x^{(\text{sort-abs})}$ by multiplying the first component by -1 in case x has an odd number of negative components. For $k, \ell \in [n]$ with $k \neq \ell$, we denote the polyhedral relation $\mathrm{R}_{-\mathbf{e}_k - \mathbf{e}_\ell, 0} \circ \mathrm{R}_{\mathbf{e}_k - \mathbf{e}_\ell, 0}$ by $E_{k,\ell}$.

Proposition 7. *If $\mathcal{R}$ is a polyhedral relation $E_{n-1,n} \circ \cdots \circ E_{1,2} \circ \mathrm{T}_{k_r,\ell_r} \circ \ldots \circ \mathrm{T}_{k_1,\ell_1}$, where $(k_1,\ell_1), \ldots, (k_r,\ell_r) \in [n] \times [n]$ is a sorting network, and $P \subseteq \mathbb{R}^n$ is a polytope with $x^{(\Phi_{D_n})} \in P$ for each vertex v of P, then we have $\mathcal{R}(P) = \Pi_{D_n}(P)$.*

Proof. With $Q = \Pi_{D_n}(P)$, the first condition of Theorem 1 is satisfied. Let us denote by $\{-,+\}^n_{\text{even}}$ the set of all $\epsilon \in \{-,+\}^n$ with an even number of components equal to minus. Then, we have $Q = \operatorname{conv}(W)$ with

$$W = \{\gamma.\epsilon.v : \gamma \in \mathfrak{S}(n), \epsilon \in \{-,+\}^n_{\text{even}}, v \text{ vertex of } P\}\,.$$

For $k, \ell \in [n]$ with $k \neq \ell$, we define $\eta^\star_{k,\ell} = \varrho^{\star(\mathrm{H}^\le(\mathbf{e}_k - \mathbf{e}_\ell, 0))} \circ \varrho^{\star(\mathrm{H}^\le(-\mathbf{e}_k - \mathbf{e}_\ell, 0))}$. For each $y \in \mathbb{R}^n$, the vector $\eta^\star_{k,\ell}(y)$ is the vector $y' \in \{y, \tau_{k,\ell}(y), \rho_{k,\ell}(y), \rho_{k,\ell}(\tau_{k,\ell}(y))\}$ with $|y'_k| \le y'_\ell$, where $\rho_{k,\ell}(y)$ arises from y by multiplying both components k and ℓ by -1. As, for every $w \in W$ with $w = \gamma.\epsilon.v$ for some vertex v of P and $\gamma \in \mathfrak{S}(n)$, $\epsilon \in \{-,+\}^n_{\text{even}}$, we have

$$\tau^\star_{k_1,\ell_1} \circ \cdots \circ \tau^\star_{k_r,\ell_r} \circ \eta^\star_{1,2} \circ \cdots \circ \eta^\star_{n-1,n}(w) = w^{(\Phi_{D_n})} = v^{(\Phi_{D_n})} \in P\,,$$

also the second condition of Theorem 1 is satisfied. Hence the claim follows.

And again, similarly to the cases A_{n-1} and B_n, we derive the following result from Proposition 7 and Remark 2.

Theorem 6. *For each polytope $P \subseteq \mathbb{R}^n$ with $v^{(\Phi_{D_n})}(v) \in P$ for each vertex v of P there is an extended formulation of $\Pi_{D_n}(P)$ with $n' + \mathrm{O}(n \log n)$ variables and $f' + \mathrm{O}(n \log n)$ inequalities, whenever P admits an extended formulation with n' variables and f' inequalities.*

If we restrict attention to the polytopes $P = \{(-1, 1, \ldots, 1)\} \subseteq \mathbb{R}^n$ and $P = \{(1, 1, \ldots, 1)\} \subseteq \mathbb{R}^n$, then we can remove the reflection relations $T_{i_1,j_1}, \ldots, T_{i_r,j_r}$ from the construction in Proposition 7. Thus, we obtain extended formulations with $2(n-1)$ variables and $4(n-1)$ inequalities of the convex hulls of all vectors in $\{-1, +1\}^n$ with an odd respectively even number of ones. Thus, applying the affine transformation of $\mathbb{R}^n$ given by $y \mapsto \frac{1}{2}(\mathbf{1} - y)$, we derive extended formulations with $2(n-1)$ variables and $4(n-1)$ inequalities for the *parity polytopes* $\mathrm{conv}\{v \in \{0,1\}^n : \sum_i v_i \text{ odd}\}$ and $\mathrm{conv}\{v \in \{0,1\}^n : \sum_i v_i \text{ even}\}$, respectively (reproving a result by Carr and Konjevod [3]).

4.2 Huffman Polytopes

A vector $v \in \mathbb{R}^n$ (with $n \geq 2$) is a *Huffman-vector* if there is a rooted binary tree with n leaves (all non-leaf nodes having two children) and a labeling of the leaves by $1, \ldots, n$ such that, for each $i \in [n]$, the number of arcs on the path from the root to the leaf labelled i equals v_i. Let us denote by $\mathrm{V}^n_{\mathrm{huff}}$ the set of all Huffman-vectors in $\mathbb{R}^n$, and by $\mathrm{P}^n_{\mathrm{huff}} = \mathrm{conv}(\mathrm{V}^n_{\mathrm{huff}})$ the *Huffman polytope*. Note that currently no linear description of $\mathrm{P}^n_{\mathrm{huff}}$ in $\mathbb{R}^n$ is known. In fact, it seems that such descriptions are extremely complicated. For instance, Nguyen, Nguyen, and Maurras [11] proved that $\mathrm{P}^n_{\mathrm{huff}}$ has $(\Omega(n))!$ facets.

It is easy to see that Huffman-vectors and -polytopes have the following properties.

Observation 1

1. *For each $\gamma \in \mathfrak{S}(n)$, we have $\gamma.\mathrm{V}^n_{\mathrm{huff}} = \mathrm{V}^n_{\mathrm{huff}}$.*
2. *For each $v \in \mathrm{V}^n_{\mathrm{huff}}$ there are at least two components of v equal to $\max\{v_k : k \in [n]\}$.*
3. *For each $v \in \mathrm{V}^n_{\mathrm{huff}}$ ($n \geq 3$) and $v_i = v_j = \max\{v_k : k \in [n]\}$ for some pair $i < j$, we have*

$$(v_1, \ldots, v_{i-1}, v_i - 1, v_{i+1}, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n) \in \mathrm{V}^{n-1}_{\mathrm{huff}} .$$

4. *For each $w' \in \mathrm{V}^{n-1}_{\mathrm{huff}}$ ($n \geq 3$), we have $(w'_1, \ldots, w'_{n-2}, w'_{n-1} + 1, w'_{n-1} + 1) \in \mathrm{V}^n_{\mathrm{huff}}$.*

For $n \geq 3$, let us define the embedding

$$P^{n-1} = \{(x_1, \ldots, x_{n-2}, x_{n-1} + 1, x_{n-1} + 1) : (x_1, \ldots, x_{n-1}) \in \mathrm{P}^{n-1}_{\mathrm{huff}}\}$$

of $\mathrm{P}^{n-1}_{\mathrm{huff}}$ into $\mathbb{R}^n$.

Proposition 8. *If $\mathcal{R} \subseteq \mathbb{R}^n \times \mathbb{R}^n$ is the polyhedral relation*

$$\mathrm{T}_{1,2} \circ \mathrm{T}_{2,3} \circ \cdots \circ \mathrm{T}_{n-2,n-1} \circ \mathrm{T}_{n-1,n} \circ \mathrm{T}_{1,2} \circ \mathrm{T}_{2,3} \circ \cdots \circ \mathrm{T}_{n-3,n-2} \circ \mathrm{T}_{n-2,n-1} , \quad (7)$$

then we have $\mathcal{R}(P^{n-1}) = \mathrm{P}^n_{\mathrm{huff}}$.

Proof. With $P = P^{n-1}$ and $Q = \mathrm{P}^n_{\mathrm{huff}}$, the first condition of Theorem 1 is obviously satisfied (due to parts (1) and (4) of Observation 1). We have $Q = \mathrm{conv}(W)$ with $W = \mathrm{V}^n_{\mathrm{huff}}$. Furthermore, for every $w \in W$ and $x = \tau^\star(w)$ with

$$\tau^\star = \tau^\star_{n-2,n-1} \circ \tau^\star_{n-3,n-2} \circ \cdots \circ \tau^\star_{2,3} \circ \tau^\star_{1,2} \circ \tau^\star_{n-1,n} \circ \tau^\star_{n-2,n-1} \circ \cdots \circ \tau^\star_{2,3} \circ \tau^\star_{1,2}\,, \quad (8)$$

we have $x_n = x_{n-1} = \max\{w_i : i \in [n]\}$, hence part (3) of Observation 1 (with $i = n-1$ and $j = n$) implies $\tau^\star(w) \in P^{n-1}$. Therefore, the claim follows by Theorem 1.

From Remark 2 we thus obtain an extended formulation for $\mathrm{P}^n_{\mathrm{huff}}$ with $n' + 2n - 3$ variables and $f' + 4n - 6$ inequalities, provided we have an extended formulation for $\mathrm{P}^{n-1}_{\mathrm{huff}}$ with n' variables and f' inequalities. As $\mathrm{P}^2_{\mathrm{huff}}$ is a single point, we thus can establish inductively the following result.

Corollary 2. *There are extended formulations of* $\mathrm{P}^n_{\mathrm{huff}}$ *with* $\mathrm{O}(n^2)$ *variables and inequalities.*

Actually, one can reduce the size of the extended formulation of $\mathrm{P}^n_{\mathrm{huff}}$ to $\mathrm{O}(n \log n)$. In order to indicate the necessary modifications, let us denote by Θ_k the sequence

$$(k-2,k-1),(k-3,k-2),\ldots,(2,3),(1,2),(k-1,k),(k-2,k-1),\ldots,(2,3),(1,2)$$

of pairs of indices used (with $k = n$) in (7) and (8). For every sequence

$$\Theta = ((i_1,j_1),\ldots,(i_r,j_r))$$

of pairs of pairwise different indices, we define $\tau^\star_\Theta = \tau^\star_{i_1,j_1} \circ \cdots \circ \tau^\star_{i_r,j_r}$ (thus, $\tau^\star$ in (8) equals $\tau^\star_{\Theta_n}$). Furthermore, we denote by $\eta_k : \mathbb{R}^k \to \mathbb{R}^{k-1}$ (for $k \geq 3$) the linear map defined via $\eta_k(y) = (y_1,\ldots,y_{k-2},y_{k-1}-1)$ for all $y \in \mathbb{R}^k$. The crucial property for the above construction to work is that the following holds for every $v \in \mathrm{V}^n_{\mathrm{huff}}$ and $k \in \{3,\ldots,n\}$: The vector

$$x = \tau^\star_{\Theta_k} \circ \eta_{k+1} \circ \tau^\star_{\Theta_{k+1}} \circ \cdots \circ \eta_n \circ \tau^\star_{\Theta_n}(v)$$

satisfies $x_{k-1} = x_k = \max\{x_i : i \in [k]\}$. It turns out that this property is preserved when replacing the sequence Θ_n by an arbitrary sorting network (e.g. of size $\mathrm{O}(n \log n)$, see Section 4.1) and, for $k \in \{3,\ldots,n-1\}$, the sequence Θ_k (of length $2k-3$) by the sequence

$$(i^k_2,i^k_1),(i^k_3,i^k_2),\ldots,(i^k_{r(k)},i^k_{r(k)-1}),(i^k_{r(k)-1},i^k_{r(k)-2}),\ldots,(i^k_3,i^k_2),(i^k_2,i^k_1)$$

with $i^k_1 = k$, $i^k_2 = k-1$, $i^k_\ell = i^k_{\ell-1} - 2^{\ell-3}$ for all $\ell \geq 3$, and $r(k)$ being the maximal ℓ with $i^k_\ell \geq 1$. As $r(k)$ is bounded by $\mathrm{O}(\log k)$ we obtain the following theorem, whose detailed proof will be included in the full version of the paper.

Theorem 7. *There are extended formulations of* $\mathrm{P}^n_{\mathrm{huff}}$ *with* $\mathrm{O}(n \log n)$ *variables and inequalities.*

5 Conclusions

We hope to have demonstrated that and how the framework of reflection relations extends the currently available toolbox for constructing extended formulations. We conclude with briefly mentioning two directions for future research.

One of the most interesting questions in this context seems to be that for other polyhedral relations that can be useful for constructing extended formulations. In particular, what other types of affinely generated polyhedral relations are there?

The reflections we referred to are reflections at hyperplanes. It would be of great interest to find tools to deal with reflections at lower dimensional subspaces as well. This, however, seems to be much harder. In particular, it is unclear whether some concept similar to that of polyhedral relations can help here.

Acknowledgements. We thank Samuel Fiorini, Michel Goemans, and Günter Rote for valuable hints and discussions.

References

1. Ajtai, M., Komlós, J., Szemerédi, E.: Sorting in $c \log n$ parallel steps. Combinatorica 3(1), 1–19 (1983)
2. Ben-Tal, A., Nemirovski, A.: On polyhedral approximations of the second-order cone. Math. Oper. Res. 26(2), 193–205 (2001)
3. Carr, R.D., Konjevod, G.: Polyhedral combinatorics. In: Greenberg, H. (ed.) Tutorials on emerging methodologies and applications in Operations Research, ch. 2, Springer, Heidelberg (2004)
4. Conforti, M., Cornuéjols, G., Zambelli, G.: Extended formulations in combinatorial optimization. 4OR 8(1), 1–48 (2010)
5. Fomin, S., Reading, N.: Root systems and generalized associahedra. In: Geometric combinatorics. IAS/Park City Math. Ser., vol. 13, pp. 63–131. AMS, Providence (2007)
6. Goemans, M.: Smallest compact formulation for the permutahedron, http://www-math.mit.edu/stringgoemans/publ.html
7. Humphreys, J.E.: Reflection groups and Coxeter groups. Cambridge Studies in Advanced Mathematics, vol. 29. Cambridge University Press, Cambridge (1990)
8. Kaibel, V., Loos, A.: Branched polyhedral systems. In: Eisenbrand, F., Shepherd, F. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 177–190. Springer, Heidelberg (2010)
9. Kaibel, V., Pashkovich, K., Theis, D.O.: Symmetry matters for the sizes of extended formulations. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 135–148. Springer, Heidelberg (2010)
10. Kipp Martin, R., Rardin, R.L., Campbell, B.A.: Polyhedral characterization of discrete dynamic programming. Oper. Res. 38(1), 127–138 (1990)
11. Nguyen, V.H., Nguyen, T.H., Maurras, J.-F.: On the convex hull of Huffman trees. Electronic Notes in Discrete Mathematics 36, 1009–1016 (2010)
12. Queyranne, M.: Structure of a simple scheduling polyhedron. Math. Programming 58(2, Ser. A), 263–285 (1993)
13. Wolsey, L.A.: Personal communication
14. Yannakakis, M.: Expressing combinatorial optimization problems by linear programs. J. Comput. System Sci. 43(3), 441–466 (1991)

Integrality Gaps of Linear and Semi-Definite Programming Relaxations for Knapsack

Anna R. Karlin[1,⋆], Claire Mathieu[2], and C. Thach Nguyen[1,⋆]

[1] University of Washington
{karlin,ncthach}@cs.washington.edu
[2] Brown University
claire@cs.brown.edu

Abstract. In this paper, we study the integrality gap of the Knapsack linear program in the Sherali-Adams and Lasserre hierarchies. First, we show that an integrality gap of $2-\epsilon$ persists up to a linear number of rounds of Sherali-Adams, despite the fact that Knapsack admits a fully polynomial time approximation scheme [24,30]. Second, we show that the Lasserre hierarchy closes the gap quickly. Specifically, after t rounds of Lasserre, the integrality gap decreases to $t/(t-1)$. This answers the open question in [9]. Also, to the best of our knowledge, this is the first positive result that uses more than a small number of rounds in the Lasserre hierarchy. Our proof uses a decomposition theorem for the Lasserre hierarchy, which may be of independent interest.

1 Introduction

Many approximation algorithms work in two phases: first, solve a linear programming (LP) or semi-definite programming (SDP) relaxation; then, round the fractional solution to obtain a feasible integer solution to the original problem. This paradigm is amazingly powerful; in particular, under the unique game conjecture, it yields the best possible ratio for MaxCut and a wide variety of other problems, see e.g. [33].

However, these algorithms have a limitation. Since they are usually analyzed by comparing the value of the output to that of the fractional solution, we cannot generally hope to get a better approximation ratio than the integrality gap of the relaxation. Furthermore, for any given combinatorial optimization problem, there are many possible LP/SDP relaxations, and it is difficult to determine which relaxations have the best integrality gaps.

This has lead to efforts to provide systematic procedures for constructing a sequence of increasingly tight mathematical programming relaxations for 0-1 optimization problems. A number of different procedures of this type have been proposed: by Lovász and Schrijver [31], Sherali and Adams [37], Balas, Ceria and Cornuejols [5], Lasserre [28,27] and others. While they differ in the details, they

⋆ The first and third authors were supported by NSF Grant CCF-0635147 and a Yahoo! Faculty Research Grant.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 301–314, 2011.

all operate in a series of rounds starting from an LP or SDP relaxation, eventually ending with an exact integer formulation. The strengthened relaxation after t rounds can typically be solved in $n^{O(t)}$ time and, roughly, satisfies the property that the values of any t variables in the original relaxation can be expressed as the projection of a convex combination of integer solutions.

A major line of research in this area has focused on understanding the strengths and limitations of these procedures. Of particular interest to our community is the question of how the integrality gaps for interesting combinatorial optimization problems evolve through a series of rounds of one of these procedures. On the one hand, if the integrality gaps of successive relaxations drop sufficiently fast, there is the potential for an improved approximation algorithm (see [15, 16, 7, 10] for example). On the other hand, a large integrality gap persisting for a large, say logarithmic, number of rounds rules out (unconditionally) a very wide class of efficient approximation algorithms, namely those whose output is analyzed by comparing it to the value of a class of LP/SDP relaxations. This implicitly contains most known sophisticated approximation algorithms for many problems including SparsestCut and MaximumSatisifiability. Indeed, serveral very strong negative results of this type have been obtained (see [2, 1, 11, 13, 19, 32, 35, 36, 34, 20, 21, 8] and others). These are also viewed as lower bounds of approximability in certain restricted models of computation.

How strong are these restricted models of computation? In other words, how much do lower bounds in these models tell us about the intrinsic hardness of the problems studied? To explore this question, we focus on one problem that is well-known to be "easy" from the viewpoint of approximability: Knapsack. We obtain the following results:

- We show that an integrality gap close to 2 persists up to a linear number of rounds of Sherali-Adams. (The integrality gap of the natural LP is 2.)
  This is interesting since Knapsack has a fully polynomial time approximation scheme [24, 30]. This confirms and amplifies what has already been observed in other contexts (e.g [13]): the Sherali-Adams restricted model of computation has serious weaknesses: a lower bound in this model does not necessarily imply that it is difficult to get a good approximation algorithm.
- We show that Lasserre's hierarchy closes the gap quickly. Specifically, after t rounds of Laserre, the integrality gap decreases to $t/(t-1)$.
  It is known that a few rounds of Lasserre can yield better relaxations. For example, two rounds of Lasserre applied to the MaxCut LP yields an SDP that is at least as strong as that used by Goemans and Williamson to get the best known approximation algorithm, and the SDP in [3] which leads to the best known approximation algorithm for SparsestCut can be obtained by three rounds of Lasserre. However, to the best of our knowledge, this is the first positive result for more than a constant number of rounds in the Lasserre hierarchy.

Our results also answer the open question in [9], which asks for the performance of lift-and-project methods for Knapsack.

1.1 Related Work

Many known approximation algorithms can be recognized in hindsight as starting from a natural relaxation and strengthening it using a couple of levels of lift-and-project. The original hope [2] had been to use lift and project systems as a systematic approach to designing novel algorithms with better approximation ratios. Instead, the last few years have mostly seen the emergence of a multitude of lower bounds. Indeed, lift and project systems have been studied mostly for well known difficult problems: MaxCut [13,17,36], SparsestCut [13,14], VertexCover [1,2,12,19,20,23,36,38], MaximumAcyclicSubgraph [13], CSP [35,39], and more.

The Knapsack problemhas a fully polynomial time approximation scheme [24, 30]. The natural LP relaxation (to be stated in full detail in the next section) has an integrality gap of $2-\epsilon$ [25]. Although we are not aware of previous work on using the lift and project systems for Knapsack, the problem of strengthening the LP relaxation via addition of well-chosen inequalities has been much the object of much interest in the past in the mathematical programming community, as stronger LP relaxations are extremely useful to speed up branch-and-bound heuristics. The knapsack polytope was studied in detail by Weismantel [40]. Valid inequalities were studied in [4,41,6]. In particular, whenever S is a minimal set (w.r.to inclusion) that does not fit in the knapsack, then $\sum_{S\cup\{j:\forall i\in S, w_j\geq w_i\}} x_j \leq |S|-1$ is a valid inequality. Generalizations and variations were also studied in [18, 22, 42]. In [9], Bienstock formulated LP with arbitrary small integrality gaps for Knapsack using "structural disjunctions". As mentioned earlier, this paper poses the question of analyzing lift-and-project methods for Knapsack. Our results answer this question.

Our results also confirm the indication from [26, 34] for example that the Sherali-Adams lift and project is not powerful enough to be an indicator of the hardness of problems. On the other hand, little is know about the Lasserre hierarchy, as the first negative results were about k-CSP [35, 39]. Our positive result leaves open the possibility that the Lasserre hierarchy may have promise as a tool to capture the intrinsic difficulty of problems.

2 Preliminaries

2.1 The Knapsack Problem

Our focus in this paper is on the Knapsack problem. In the Knapsack problem, we are given a set of n objects $V=[n]$ with sizes $c_1, c_2, \ldots c_n$, values $v_1, v_2, \ldots v_n$, and a capacity C. We assume that for every i, $c_i \leq C$. The objective is to select a subset of objects of maximum total value such that the total size of the objects selected does not exceed C.

The standard linear programming (LP) relaxation [25] for Knapsack is given by:

$$\max \quad \sum_{i \in V} v_i x_i \quad \text{s.t.} \quad \begin{cases} \sum_{i \in V} c_i x_i \leq C \\ 0 \leq x_i \leq 1 \qquad \forall i \in V \end{cases} \tag{1}$$

The intended intepretation of an integral solution of this LP is obvious: $x_i = 1$ means the object i is selected, and $x_i = 0$ means it is not. The constraint can be written as $g(x) = C - \sum_i c_i x_i \geq 0$.

Let *Greedy* denote the algorithm that puts objects in the knapsack by order of decreasing ratio v_i/c_i, stopping as soon as the next object would exceed the capacity. The following lemma is folklore.

Lemma 1. *Consider an instance (C, V) of* Knapsack *and its LP relaxation K given by (1). Then Value$(K) \leq$ Value(Greedy)$(C, V)) + \max_{i \in V} v_i$.*

2.2 The Sherali-Adams and Lasserre Hierarchies

We next review the lift-and-project hierarchies that we will use in this paper. The descriptions we give here assume that the base program is linear and mostly use the notation given in the survey paper by Laurent [29]. To see that these hierarchies apply at a much greater level of generality we refer the reader to Laurent's paper [29].

Let K be a polytope defined by a set of linear constraints $g_1, g_2, \ldots g_m$:

$$K = \{x \in [0,1]^n | g_\ell(x) \geq 0 \text{ for } \ell = 1, 2, \ldots m\}. \tag{2}$$

We are interested in optimizing a linear objective function f over the convex hull $P = \mathsf{conv}\,(K \cap \{0,1\}^n)$ of integral points in K. Here, P is the set of convex combinations of all integral solutions of the given combinatorial problem and K is the set of solutions to its linear relaxation. For example, if K is defined by (1), then P is the set of convex combinations of valid integer solutions to Knapsack.

If all vertices of K are integral then $P = K$ and we are done. Otherwise, we would like to strengthen the relaxation K by adding additional valid constraints. The Sherali-Adams (SA) and Lasserre hierarchies are two different systematic ways to construct these additional constraints. In the SA hierarchy, all the constraints added are linear, whereas Lasserre's hierarchy is stronger and introduces a set of positive semi-definite constraints. However, for consistency, we will describe both hierarchies as requiring certain submatrices to be positive semi-definite.

To this end, we first state some notations. Throughout this paper we will use $\mathcal{P}(V)$ to denote the power set of V, and $\mathcal{P}_t(V)$ to denote the collection of all subsets of V whose sizes are at most t. Also, given two sets of coordinates T and S, $T \subseteq S$ and $y \in R^S$, by $y|_T$ we denote the projection of y onto T.

Next, we review the definition of the *shift operator* between two vectors $x, y \in R^{\mathcal{P}(V)}$: $x * y$ is a vector in $R^{\mathcal{P}(V)}$ such that

$$(x * y)_I = \sum_{J \subseteq V} x_J y_{I \cup J}.$$

Lemma 2 ([29]). *The shift operator is commutative: for any vectors* $x, y, z \in R^{\mathcal{P}(V)}$*, we have* $x * (y * z) = y * (x * z)$.

A polynomial $P(x) = \sum_{I \subseteq V} a_I \prod_{i \in I} x_i$ can also be viewed as a vector indexed by subsets of V. We define the vector $P*y$ accordingly: $(P*y)_I = \sum_{J \subseteq V} a_J y_{I \cup J}$.

Finally, let $\mathcal{T}$ be a collection of subsets of V and y be a vector in $R^{\mathcal{T}}$. We denote by $M_{\mathcal{T}}(y)$ the matrix whose rows and colums are indexed by elements of $\mathcal{T}$ such that

$$(M_{\mathcal{T}}(y))_{I,J} = y_{I \cup J}.$$

The main observation is that if $x \in K \cap \{0,1\}^n$ then $(y_I) = (\prod_{i \in I} x_i)$ satisfies $M_{\mathcal{P}(V)}(y) = yy^T \succeq 0$ and $M_{\mathcal{P}(V)}(g_\ell * y) = g_\ell(x) yy^T \succeq 0$ for all constraints g_ℓ. Thus requiring principal submatrices of these two matrices to be positive semi-definite yields a relaxation.

Definition 1. *For any* $1 \leq t \leq n$*, the* t-th Sherali-Adams lifted polytope $\mathsf{SA}^t(K)$ *is the set of vectors* $y \in [0,1]^{\mathcal{P}_t(V)}$ *such that* $y_\emptyset = 1$*,* $M_{\mathcal{P}(U)}(y) \succeq 0$ *and* $M_{\mathcal{P}(W)}(g_\ell * y) \succeq 0$ *for all* ℓ *and subsets* $U, W \subseteq V$ *such that* $|U| \leq t$ *and* $|W| \leq t-1$.

We say that a point $x \in [0,1]^n$ *belongs to the* t-th Sherali-Adams polytope $\mathsf{sa}^t(K)$ *iff there exists a* $y \in \mathsf{SA}^t(K)$ *such that* $y_{\{i\}} = x_i$ *for all* $i \in [n]$.

Definition 2. *For any* $1 \leq t \leq n$*, the* t-th Lasserre lifted polytope $\mathsf{La}^t(K)$ *is the set of vectors* $y \in [0,1]^{\mathcal{P}_{2t}(V)}$ *such that* $y_\emptyset = 1$*,* $M_{\mathcal{P}_t(V)}(y) \succeq 0$ *and* $M_{\mathcal{P}_{t-1}(V)}(g_\ell * y) \succeq 0$ *for all* ℓ.

We say that a point $x \in [0,1]^n$ *belongs to the* t-th Lasserre polytope $\mathsf{la}^t(K)$ *if there exists a* $y \in \mathsf{La}^t(K)$ *such that* $y_{\{i\}} = x_i$ *for all* $i \in V$.

Note that $M_{\mathcal{P}(U)}(y)$ has at most 2^t rows and columns, which is constant for t constant, whereas $M_{\mathcal{P}_t(V)}(y)$ has $\binom{n+1}{t+1}$ rows and columns.

It is immediate from the definitions that $\mathsf{sa}^{t+1}(K) \subseteq \mathsf{sa}^t(K)$, and $\mathsf{la}^{t+1}(K) \subseteq \mathsf{la}^t(K)$ for all $1 \leq t \leq n-1$. Sherali and Adams [37] show that $\mathsf{sa}^n(K) = P$, and Lasserre [28,27] show that $\mathsf{la}^n(K) = P$. Thus, the sequences

$$K \supseteq \mathsf{sa}^1(K) \supseteq \mathsf{sa}^2(K) \supseteq \cdots \supseteq \mathsf{sa}^n(K) = P$$
$$K \supseteq \mathsf{la}^1(K) \supseteq \mathsf{la}^2(K) \supseteq \cdots \supseteq \mathsf{la}^n(K) = P$$

define hierarchies of polytopes that converge to P. Furthermore, the Lasserre hierarchy is stronger than the Sherali-Adams hierarchy: $\mathsf{la}^t(K) \subseteq \mathsf{sa}^t(K)$ for all $1 \leq t \leq n$. In this paper, we show that for the Knapsack problem, the Lasserre hierarchy is strictly stronger.

2.3 Proof Overview

Consider instances of Knapsack where at most $k-1$ objects can be put into the knapsack. Examples are instances where all objects have sizes greater than C/k.

Such instances are easy: they can be solved by going through all subsets of at most $k-1$ objects. We ask if SA and La "realize" this.

It turns out that SA does not. In fact, our lower bound instances fall into this category. For $t \geq k$, SA^t does require that $y_I = 0$ if I does not fit in the knapsack; in some senses, this means the fractional solution y has to be a combination of integral solutions. However, SA^t has very few constraints on the "interaction" of these solutions, thus fails to enforce the combination to be *convex*. In fact, the solution in Section 3 can be viewed as a combination of feasible integral solutions, but the coefficients of this combination do not sum to 1.

On the other hand, La^k handles these special instances. La^k also requires that $y_I = 0$ for any set I that does not fit in the knapsack. More importantly, if we extend y so that $y_I = 0$ for $|I| > k$, then the two constraints $M_{\mathcal{P}(V)}(y) \succeq 0$ and $M_{\mathcal{P}_k(V)}(y) \succeq 0$ are essentially the same, since the former matrix is the latter one padded by 0's. The former constraint requires y to be a convex combination of integral solutions while the latter is what La^k enforces.

Now consider a general Knapsack instance K and let $y \in \mathsf{La}^k(K)$. Let OPT be the optimal value of K, and L be the set of objects whose values are greater than OPT/k. Then no subset of more than $k-1$ objects in L can be put into the knapsack. Thus, y is a convex combination of vectors which are integral on $\mathcal{P}_k(L)$. If these (fractional) vectors are feasible for the original Knapsack LP, then by Lemma 1, the value of each can be bounded by $OPT + \max_{i \notin L} v_i$, which is at most $\frac{k+1}{k} OPT$. Hence so is the value of y.

Proving that these vectors are feasible for the original Knapsack LP turns out to be very technical and Section 4 is dedicated to it. The section proves a stronger fact that any $y \in \mathsf{La}^t(K)$ can be written as a convex combination of vectors which are intergral on $\mathcal{P}_t(L)$ and feasible in $\mathsf{La}^{t-k}(K)$). Section 5 then finishes the analysis of La for Knapsack.

3 Lower Bound for the Sherali-Adams Hierarchy for Knapsack

In this section, we show that the integrality gap of the t-th level of the Sherali-Adams hierarchy for Knapsack is close to 2. This lower bound even holds for the *uniform*[1] Knapsack problem, in which $v_i = c_i = 1$ for all i.

Theorem 1. *For every $\epsilon, \delta > 0$, the integrality gap at the t-th level of the Sherali-Adams hierarchy for* Knapsack *where $t \leq \delta n$ is at least $(2-\epsilon)(1/(1+\delta))$.*

Proof. (Sketch) Consider the instance K of Knapsack with n objects where $c_i = v_i = 1$ for all $i \in V$ and capacity $C = 2(1-\epsilon)$. Then the optimal integer value is 1. On the other hand, we claim that the vector y where $y_\emptyset = 1$, $y_{\{i\}} = C/(n+(t-1)(1-\epsilon))$ and $y_I = 0$ for all $|I| > 1$ is in $\mathsf{SA}^t(K)$. Thus, the integrality gap of the tth round of Sherali-Adams is at least $Cn/(n+(t-1)(1-\epsilon))$, which is at least $(2-\epsilon)(1/(1+\delta))$ when $t \leq \delta n$.

[1] This problem is also known as *Unweighted Knapsack* or *Subset Sum*.

4 A Decomposition Theorem for the Lasserre Hierarchy

In this section, we develop the machinery we will need for our Lasserre upper bounds. It turns out that it is more convenient to work with families (z^X) of characteristic vectors rather than directly with y. We begin with some definitions and basic properties.

Definition 3 (extension). *Let $\mathcal{T}$ be a collection of subsets of V and let y be a vector indexed by sets of $\mathcal{T}$. We define the* extension *of y to be the vector y', indexed by all subsets of V, such that y'_I equals y_I if $I \in \mathcal{T}$ and equals 0 otherwise.*

Definition 4 (characteristic polynomial). *Let S be a subset of V and X a subset of S. We define the* characteristic polynomial *P^X of X with respect to S as*

$$P^X(x) = \prod_{i \in X} x_i \prod_{j \in S \setminus X} (1 - x_j) = \sum_{J : X \subseteq J \subseteq S} (-1)^{|J \setminus X|} \prod_{i \in J} x_i.$$

Lemma 3 (inversion formula). *Let y' be a vector indexed by all subsets of V. Let S be a subset of V and, for each X subset of S, let $z^X = P^X * y'$:*

$$z^X_I = \sum_{J : X \subseteq J \subseteq S} (-1)^{|J \setminus X|} y'_{I \cup J}.$$

Then $y' = \sum_{X \subseteq S} z^X$.

Lemma 4. *Let y' be a vector indexed by all subsets of V, S be a subset of V and X be a subset of S. Then*

$$\begin{cases} z^X_I = z^X_{I \setminus X} & \text{for all } I \\ z^X_I = z^X_\emptyset & \text{if } I \subseteq X \\ z^X_I = 0 & \text{if } I \cap (S \setminus X) \neq \emptyset \end{cases}$$

Proof. Let $I' = I \setminus X$ and $I'' = I \cap X$. Using the definition of z^X_I and noticing that $X \cup I'' = X$ yields $z^X_I = z^X_{I'}$. This immediately implies that for $I \subseteq X$, $z^X_I = z^X_\emptyset$.

Finally, consider a set I that intersects $S \setminus X$ and let $i \in I \cap (S \setminus X)$. In the definition of z^X_I, we group the terms of the sum into pairs consisting of J such that $i \notin J$ and of $J \cup \{i\}$. Since $I = I \cup \{i\}$, we obtain:

$$\sum_{J : X \subseteq J \subseteq S} (-1)^{|J \setminus X|} y'_{I \cup J} = \sum_{J : X \subseteq J \subseteq S \setminus \{i\}} \left((-1)^{|J \setminus X|} + (-1)^{|J \setminus X| + 1} \right) y'_{I \cup J} = 0.$$

Corollary 1. *Let y' be a vector indexed by all subsets of V, S be a subset of V and X be a subset of S. Let w^X be defined as $z^X / z^X_\emptyset$ if $z^X_\emptyset \neq 0$ and defined as 0 otherwise. Then, if $z^X_\emptyset \neq 0$, then $w^X_{\{i\}}$ equals 1 for elements of X and 0 for elements of $S \setminus X$.*

Definition 5 (closed under shifting). *Let S be an arbitrary subset of V and $\mathcal{T}$ be a collection of subsets of V. We say that $\mathcal{T}$ is* closed under shifting by S *if*

$$Y \in \mathcal{T} \implies \forall X \subseteq S, \quad X \cup Y \in \mathcal{T}.$$

The following lemma generalizes Lemma 5 in [29]. It proves that the positive-semidefinite property carries over from y to (z^X).

Lemma 5. *Let S be an arbitrary subset of V and $\mathcal{T}$ be a collection of subsets of V that is closed under shifting by S. Let y be a vector indexed by sets of $\mathcal{T}$. Then*

$$M_{\mathcal{T}}(y) \succeq 0 \implies \forall X \subseteq S, \quad M_{\mathcal{T}}(z^X) \succeq 0.$$

In the rest of the section, we prove a decomposition theorem for the Lasserre hierarchy, which allows us to "divide" the action of the hierarchy and think of it as using the first few rounds on some subset of variables, and the other rounds on the rest. We will use this theorem to prove that the Lasserre hierarchy closes the gap for the Knapsack problem in the next section.

Theorem 2. *Let $t > 1$ and $y \in \mathsf{La}^t(K)$. Let $k < t$ and S be a subset of V and such that*

$$|I \cap S| \geq k \implies y_I = 0. \tag{3}$$

Consider the projection $y|_{\mathcal{P}_{2t-2k}(V)}$ of y to the coordinates corresponding to subsets of size at most $2t-2k$ of V. Then there exist subsets $X_1, X_2, \ldots, X_m$ of S such that $y|_{\mathcal{P}_{2t-2k}(V)}$ is a convex combination of vectors w^{X_i} with the following properties:

- $w^{X_i}_{\{j\}} = \begin{cases} 1 & \text{if } j \in X_i \\ 0 & \text{if } j \in S \setminus X_i; \end{cases}$
- $w^{X_i} \in \mathsf{La}^{t-k}(K)$*; and*
- *if K_i is obtained from K by setting $x_j = w^{X_i}_{\{j\}}$ for $j \in S$, then $w^{X_i}|_{\mathcal{P}_{2t-2k}(V \setminus S)} \in \mathsf{La}^{t-k}(K_i)$.*

To prove Theorem 2, we will need a couple more lemmas. In the first one, using assumption (3), we extend the positive semi-definite properties from y to y', and then, using Lemma 5, from y' to z^X.

Lemma 6. *Let t, y, S, k be defined as in Theorem 2, and y' be the extension of y. Let $\mathcal{T}_1 = \{A \text{ such that } |A \backslash S| \leq t-k\}$, and $\mathcal{T}_2 = \{B \text{ such that } |B \backslash S| < t-k\}$. Then for all $X \subseteq S$, $M_{\mathcal{T}_1}(z^X) \succeq 0$ and, for all ℓ, $M_{\mathcal{T}_2}(g_\ell * z^X) \succeq 0$.*

Proof. We will first prove that $M_{\mathcal{T}_1}(y') \succeq 0$ and, for all ℓ, $M_{\mathcal{T}_2}(g_\ell * y') \succeq 0$. Order the columns and rows of $M_{\mathcal{T}_1}(y')$ by subsets of non-decreasing size. By definition of $\mathcal{T}_1$, any $I \in \mathcal{T}_1$ of size at least t must have $|I \cap S| \geq k$, and so $y'_I = 0$. Thus

$$M_{\mathcal{T}_1}(y') = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix},$$

where M is a principal submatrix of $M_{\mathcal{P}_t(V)}(y)$.Thus $M \succeq 0$, and so $M_{\mathcal{T}_1}(y') \succeq 0$.

Similarly, any $J \in \mathcal{T}_2$ of size at least $t-1$ must have $|J \cup \{i\} \cap S| \geq k$ for every i as well as $|J \cap S| \geq k$, and so, by definition of $g_\ell * y'$ we must have $(g_\ell * y')_J = 0$. Thus

$$M_{\mathcal{T}_2}(g_\ell * y') = \begin{pmatrix} N & 0 \\ 0 & 0 \end{pmatrix},$$

where N is a principal submatrix of $M_{\mathcal{P}_{t-1}(V)}(g_\ell * y)$. Thus $N \succeq 0$, and so $M_{\mathcal{T}_2}(g_\ell * y') \succeq 0$.

Observe that $\mathcal{T}_1$ is closed under shifting by S. By definition of z^X and Lemma 5, we thus get $M_{\mathcal{T}_1}(z^X) \succeq 0$.

Similarly, observe that $\mathcal{T}_2$ is also closed under shifting by S. By Lemma 2, we have $g_\ell * (P^X * y') = P^X * (g_\ell * y')$, and so by Lemma 5 again we get $M_{\mathcal{T}_2}(g_\ell * z^X) \succeq 0$.

Lemma 7. *Let t, y, S, k be defined as in Theorem 2, and y' be the extension of y. Then for any $X \subseteq S$:*

1. $z^X_\emptyset \geq 0$.
2. *If $z^X_\emptyset = 0$ then $z^X_I = 0$ for all $|I| \leq 2t - 2k$.*

Proof. Let $\mathcal{T}_1$ be defined as in Lemma 6. By Lemma 6 $M_{\mathcal{T}_1}(z^X) \succeq 0$ and $z^X_\emptyset$ is a diagonal element of this matrix, hence $z^X_\emptyset \geq 0$.

For the second part, start by considering $J \subseteq V$ of size at most $t-k$. Then $J \in \mathcal{T}_1$, and so the matrix $M_{\{\emptyset, J\}}(z^X)$ is a principal submatrix of $M_{\mathcal{T}_1}(z^X)$, hence is also positive semidefinite. Since $z^X_\emptyset = 0$,

$$M_{\{\emptyset,J\}}(z^X) = \begin{pmatrix} 0 & z^X_J \\ z^X_J & z^X_J \end{pmatrix} \succeq 0,$$

hence $z^X_J = 0$.

Now consider any $I \subseteq V$ such that $|I| \leq 2t - 2k$, and write $I = I_1 \cup I_2$ where $|I_1| \leq t - k$ and $|I_2| \leq t - k$. $M_{\{I_1, I_2\}}(z^X)$ is a principal submatrix of $M_{\mathcal{T}_1}(z^X)$, hence is also positive semidefinite. Since $z^X_{I_1} = z^X_{I_2} = 0$,

$$M_{\{I_1,I_2\}}(z^X) = \begin{pmatrix} 0 & z^X_I \\ z^X_I & 0 \end{pmatrix} \succeq 0,$$

hence $z^X_I = 0$.

We now have what we need to prove Theorem 2.

Proof (of Theorem 2). By definition, Lemma 3 and the second part of Lemma 7, we have

$$y|_{\mathcal{P}_{2t-2k}(V)} = y'|_{\mathcal{P}_{2t-2k}(V)} = \sum_{X \subseteq S} z^X|_{\mathcal{P}_{2t-2k}(V)} = \sum_{X \subseteq S} z^X_\emptyset w^X|_{\mathcal{P}_{2t-2k}(V)}.$$

By Lemma 3 and by definition of y, we have $\sum_{X \subseteq S} z^X_\emptyset = y_\emptyset = 1$, and the terms are non-negative by the first part of Lemma 7, so $y|_{\mathcal{P}_{2t-2k}(V)}$ is a convex combination of w^X's, as desired.

Consider $X \subseteq S$ such that $z^X_\emptyset \neq 0$. By Lemma 6, $M_{\mathcal{T}_1}(z^X) \succeq 0$ and $M_{\mathcal{T}_2}(g_\ell * z^X) \succeq 0$ for all ℓ, and so this also holds for their principal submatrices $M_{\mathcal{P}_{t-k}(V)}(z^X)$ and $M_{\mathcal{P}_{t-k-1}(V)}(g_\ell * z^X)$. Scaling by the positive quantity $z^X_\emptyset$, by definition of w^X this also holds for $M_{\mathcal{P}_{t-k}(V)}(w^X)$ and $M_{\mathcal{P}_{t-k-1}(V)}(g_\ell * w^X)$. In other words, $w^X|_{\mathcal{P}_{2t-2k}(V)} \in \mathsf{La}^{t-k}(K)$.

Since $M_{\mathcal{P}_{t-k}(V)}(w^{X_i}) \succeq 0$, by taking a principal submatrix, we infer that $M_{\mathcal{P}_{t-k}(V\setminus S)}(w^{X_i}) \succeq 0$. Similarly, $M_{\mathcal{P}_{t-k}(V)}(g_\ell * w^{X_i}) \succeq 0$ and so $M_{\mathcal{P}_{t-k}(V\setminus S)}(g_\ell * w^{X_i}) \succeq 0$. Let g'_ℓ be the constraint of K_i obtained from g_ℓ by setting $x_j = w^{X_i}_{\{j\}}$ for all $j \in S$. We claim that for any $I \subseteq V\setminus S$, $(g'_\ell * z^{X_i})_I = (g_\ell * z^{X_i})_I$; scaling implies that $M_{\mathcal{P}_{t-k}(V\setminus S)}(g'_\ell * w^{X_i}) = M_{\mathcal{P}_{t-k}(V\setminus S)}(g_\ell * w^{X_i})$ and we are done.

To prove the claim, let $g_\ell(x) = \sum_{j\in V} a_j x_j + b$. Then, by Corollary 1, $g'_\ell = \sum_{j\in V\setminus S} a_j x_j + (b + \sum_{j\in X_i} a_j)$. Let $I \subseteq V\setminus S$. We see that

$$(g_\ell * w^{X_i})_I - (g'_\ell * w^{X_i})_I = \sum_{j\in X_i} a_j w^{X_i}_{I\cup\{j\}} + \sum_{j\in S\setminus X_i} a_j w^{X_i}_{I\cup\{J\}} - \sum_{j\in X_i} a_j w^{X_i}_I.$$

By Lemma 4, $w^{X_i}_{I\cup\{j\}} = w^{X_i}_I$ for $j \in X_i$ and $w^{X_i}_{I\cup\{j\}} = 0$ for $j \in S\setminus X_i$. The claim follows.

5 Upper Bound for the Lasserre Hierarchy for Knapsack

In this section, we use Theorem 2 to prove that for the Knapsack problem the gap of $\mathsf{La}^t(K)$ approaches 1 quickly as t grows, where K is the LP relaxation of (1). First, we show that there is a set S such that every feasible solution in $\mathsf{La}^t(K)$ satisfies the condition of the Theorem.

Given an instance (C, V) of Knapsack, Let $OPT(C, V)$ denote the value of the optimal integral solution.

Lemma 8. *Consider an instance (C, V) of* Knapsack *and its linear programming relaxation K given by (1). Let $t > 1$ and $y \in \mathsf{La}^t(K)$. Let $k < t$ and $S = \{i \in V | v_i > OPT(C, V)/k\}$. Then:*

$$\sum_{i\in I\cap S} c_i > C \implies y_I = 0.$$

Proof. There are three cases depending on the size of I:

1. $|I| \leq t - 1$. Recall the capacity constraint $g(x) = C - \sum_{i\in V} c_i x_i \geq 0$. On the one hand, since $M_{\mathcal{P}_{t-1}(V)}(g * y) \succeq 0$, the diagonal entry $(g * y)_I$ must be non-negative. On the other hand, writing out the definition of $(g * y)_I$ and noting that the coefficients c_i are all non-negative, we infer $(g * y)_I \leq Cy_I - \left(\sum_{i\in I} c_i\right) y_I$. But by assumption, $\sum_{i\in I} c_i > C$. Thus we must have $y_I = 0$.

2. $t \leq |I| \leq 2t-2$. Write $I = I_1 \cup I_2 = I$ with $|I_1|, |I_2| \leq t-1$ and $|I_1 \cap S| \geq k$. Then $y_{I_1} = 0$. Since $M_{\mathcal{P}_t(y)} \succeq 0$, its 2-by-2 principal submatrix $M_{\{I_1,I_2\}}(y)$ must also be positive semi-definite.

$$M_{\{I_1,I_2\}}(y) = \begin{pmatrix} 0 & y_I \\ y_I & y_{I_2} \end{pmatrix},$$

and it is easy to check that we must then have $y_I = 0$.
3. $2t-1 \leq |I| \leq 2t$. Write $I = I_1 \cup I_2 = I$ with $|I_1|, |I_2| \leq t$ and $|I_1 \cap S| \geq k$. Then $y_{I_1} = 0$ since $t \leq 2t-2$ for all $t \geq 2$. By the same argument as in the previous case, we must then have $y_I = 0$.

The following theorem shows that the integrality gap of the t^{th} level of the Lasserre hierarchy for Knapsack reduces quickly when t increases.

Theorem 3. *Consider an instance (C, V) of* Knapsack *and its LP relaxation K given by (1). Let $t \geq 2$. Then*

$$\mathit{Value}(\mathsf{La}^t(K)) \leq (1 + \frac{1}{t-1})OPT,$$

and so the integrality gap at the t-th level of the Lasserre hierarchy is at most $1 + 1/(t-1)$.

Proof. Let $S = \{i \in V | v_i > OPT(C,V)/(t-1)\}$. Let $y \in \mathsf{La}^t(K)$. If $|I \cap S| \geq t-1$, then the elements of $I \cap S$ have total value greater than $OPT(C,V)$, so they must not be able to fit in the knapsack: their total capacity exceeds C, and so by Lemma 8 we have $y_I = 0$. Thus the condition of Theorem 2 holds for $k = t-1$.

Therefore, $y|_{\mathcal{P}_2(V)}$ is a convex combination of w^{X_i} with $X_i \subseteq S$, thus $\text{Value}(y) \leq \max_i \text{Value}(w^{X_i})$. By the first and third properties of the Theorem, we have:

$$\text{Value}(w^{X_i}) \leq \sum_{j \in X_i} v_j + \text{Value}(\mathsf{La}^1(K_i)).$$

By the nesting property of the Lasserre hierarchy, Lemma 1, and the definition of S,

$$\text{Value}(\mathsf{La}^1(K_i)) \leq \text{Value}(K_i) \leq OPT(C - \text{Cost}(X_i), V \setminus S)) + \frac{OPT(C,V)}{t-1}.$$

By the second property of the Theorem, w^{X_i} is in $\mathsf{La}^{t-k}(K) \subseteq K$, so it must satisfy the capacity constraint, so $\sum_{i \in X_i} c_i \leq \sum_{i \in I} c_i \leq C$, so X_i is feasible. Thus:

$$\text{Value}(y) \leq \max_{\text{feasible } X \subseteq S} \left(\sum_{j \in X} v_j + OPT(C - \text{Cost}(X), V \setminus S)) \right) + \frac{OPT(C,V)}{t-1}$$

The first expression in the right hand side is equal to $OPT(C,V)$, hence the Theorem.

6 Conclusion

We have shown that for Knapsack, an integrality gap of $2 - \epsilon$ persists up to a linear number of rounds in the Sherali-Adams hierarchy. This broadens the class of problems for which Sherali-Adams is not strong enough to capture the instrinsic difficulty of problems.

On the other hand, the positive result for Lasserre opens the posibility that lower bounds in the Lasserre hierarchy are good indicators of the intrinsic difficulty of the problems at hand. Further investigations into the performance of the Lasserre Hierarchy on other "easy" problems such as SpanningTree, BinPacking, etc. to either confirm or reject this possibility would be of interest.

One obstacle along this line is the fact that the second positive semidefinite constraint of the hierarchy ($M_{\mathcal{P}(t)V}(g_\ell * y) \succeq 0$) is notoriously hard to deal with, especially when g_ℓ contains many variables (in the lowerbounds for k-CSPs [35, 39], the authors are able to get around this by constructing vectors for only valid assignments, an approach that is possible only when all the constraints are "small".) Clearly, both lower bounds and upper bounds for the Lasserre hierarchy for problems with large constraints remain interesting to pursue.

Acknowledgement

Clare Mathieu would like to thank Eden Chlamtac for stimulating discussions.

References

1. Alekhnovich, M., Arora, S., Tourlakis, I.: Towards strong non-approximability results in the Lovász-Schrijver hierarchy. In: ACM STOC (2005)
2. Arora, S., Bollobás, B., Lovász, L., Tourlakis, I.: Proving integrality gaps without knowing the linear program. Theory of Computing 2, 19–51 (2006)
3. Arora, S., Rao, S., Vazirani, U.V.: Expander flows, geometric embeddings and graph partitioning. J. ACM 56(2) (2009)
4. Balas, E.: Facets of the Knapsack Polytope (1998)
5. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. Mathematical Programming 58, 295–324 (1993)
6. Balas, E., Zemel, E.: Facets of the knapsack polytope from minimal covers. SIAM Journal on Applied Mathematics 34, 119–148 (1978)
7. Bateni, M.H., Charikar, M., Guruswami, V.: MaxMin allocation via degree lower-bounded arborescences. In: ACM STOC (2009)
8. Benabbas, S., Magen, A.: Extending SDP integrality gaps to sherali-adams with applications to QUADRATIC PROGRAMMING and MAXCUTGAIN. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 299–312. Springer, Heidelberg (2010)
9. Bienstock, D.: Approximate formulations for 0-1 knapsack sets. Operational Research Letters 36(3), 317–320 (2008)
10. Bienstock, D., Ozbay, N.: Tree-width and the Sherali-Adams operator. Discrete Optimization 1(1), 13–21 (2004)
11. Buresh-Oppenheim, J., Galesi, N., Hoory, S., Magen, A., Pitassi, T.: Rank bounds and integrality gaps for cutting plane procedures. In: IEEE FOCS (2003)

12. Charikar, M.: On semidefinite programming relaxations for graph coloring and vertex cover. In: ACM-SIAM SODA (2002)
13. Charikar, M., Makarychev, K., Makarychev, Y.: Integrality gaps for Sherali-Adams relaxations. In: ACM STOC (2009)
14. Cheeger, J., Kleiner, B., Naor, A.: A $(\log n)^{\Omega(1)}$ integrality gap for the Sparsest Cut SDP. In: IEEE FOCS (2009)
15. Chlamtac, E.: Approximation algorithms using hierarchies of semidefinite programming relaxations. In: IEEE FOCS (2007)
16. Chlamtac, E., Singh, G.: Improved approximation guarantees through higher levels of SDP hierarchies. In: Goel, A., Jansen, K., Rolim, J.D.P., Rubinfeld, R. (eds.) APPROX and RANDOM 2008. LNCS, vol. 5171, pp. 49–62. Springer, Heidelberg (2008)
17. Fernandez de la Vega, W., Kenyon-Mathieu, C.: Linear programming relaxations of MaxCut. In: ACM-SIAM SODA (2007)
18. Escudero, L.F., Garn, A.: An o(n log n) procedure for identifying facets of the knapsack polytope. Operational Research Letters 31(3), 211–218 (2003)
19. Georgiou, K., Magen, A., Pitassi, T., Tourlakis, I.: Integrality gaps of $2 - o(1)$ for vertex cover SDPs in the Lovász-Schrijver hierarchy. In: IEEE FOCS (2007)
20. Georgiou, K., Magen, A., Tourlakis, I.: Vertex cover resists sDPs tightened by local hypermetric inequalities. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO 2008. LNCS, vol. IPCO, pp. 140–153. Springer, Heidelberg (2008)
21. Georgiou, K., Magen, A., Tulsiani, M.: Optimal sherali-adams gaps from pairwise independence. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX 2009. LNCS, vol. 5687, pp. 125–139. Springer, Heidelberg (2009)
22. Hartvigsen, D., Zemel, E.: On the computational complexity of facets and valid inequalities for the knapsack problem. Discrete Applied Math. 39, 113–123 (1992)
23. Hatami, H., Magen, A., Markakis, E.: Integrality gaps of semidefinite programs for vertex cover and relations to ℓ_1 embeddability of negative type metrics. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) RANDOM 2007 and APPROX 2007. LNCS, vol. 4627, pp. 164–179. Springer, Heidelberg (2007)
24. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. Journal of the ACM (1975)
25. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack problems. Springer, Heidelberg (2003)
26. Khot, S., Saket, R.: SDP integrality gaps with local ℓ_1-embeddability. In: IEEE FOCS (2009)
27. Lasserre, J.B.: An explicit exact SDP relaxation for nonlinear 0-1 programs. In: Aardal, K., Gerards, B. (eds.) IPCO 2001. LNCS, vol. 2081, p. 293. Springer, Heidelberg (2001)
28. Lasserre, J.B.: Global optimization with polynomials and the problem of moments. SIAM Journal on Optimization 11, 796–817 (2001)
29. Laurent, M.: A comparison of the Sherali-Adams, Lovász-schrijver and Lasserre relaxations for 0-1 programming. Mathematics of Operations Research 28, 470–496 (2003)
30. Lawler, E.L.: Fast approximation algorithms for knapsack problems. In: IEEE FOCS (1997)
31. Lovász, L., Schrijver, A.: Cones of matrices and set-functions and 0-1 optimization. SIAM Journal on Optimization 1, 166–190 (1991)
32. Pitassi, T., Segerlind, N.: Exponential lower bounds and integrality gaps for tree-like Lovász-Schrijver procedures. In: ACM-SIAM SODA (2009)

33. Raghavendra, P.: Optimal algorithms and inapproximability results for every CSP? In: ACM STOC (2008)
34. Raghavendra, P., Steurer, D.: Integrality gaps for strong SDP relaxations of unique games. In: IEEE FOCS (2009)
35. Schoenebeck, G.: Linear level Lasserre lower bounds for certain k-csps. In: IEEE FOCS (2008)
36. Schoenebeck, G., Trevisan, L., Tulsiani, M.: Tight integrality gaps for Lovász-Schrijver SDP relaxations of vertex cover and max cut. In: ACM STOC, pp. 302–310 (2007)
37. Sherali, H.D., Adams, W.P.: A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM Journal on Discrete Mathematics 3, 411–430 (1990)
38. Tourlakis, I.: New lower bounds for vertex cover in the Lovász-Schrijver hierarchy. In: IEEE CCC (2006)
39. Tulsiani, M.: CSP gaps and reductions in the Lasserre hierarchy. In: ACM STOC (2009)
40. Weismantel, R.: On the 0/1 knapsack polytope. Mathematical Programming 77, 49–68 (1997)
41. Wolsey, L.A.: Valid inequalities for 0-1 knapsacks and mips with generalised upper bound constraints. Discrete Applied Mathematics 29(2-3), 251–261 (1990)
42. Zemel, E.: Easily computable facets of the knapsack problem. Mathematics of Operations Research 14, 760–774 (1989)

Degree Bounded Forest Covering

Tamás Király[1,⋆] and Lap Chi Lau[2,⋆⋆]

[1] MTA-ELTE Egerváry Research Group, Deptartment of Operations Research
Eötvös Loránd University, Budapest
tkiraly@cs.elte.hu
[2] Department of Computer Science and Engineering
The Chinese University of Hong Kong
chi@cse.cuhk.edu.hk

Abstract. We prove that for an undirected graph with arboricity at most $k+\epsilon$, its edges can be decomposed into k forests and a subgraph with maximum degree $\lceil \frac{k\epsilon+1}{1-\epsilon} \rceil$. The problem is solved by a linear programming based approach: we first prove that there exists a fractional solution to the problem, and then use a result on the degree bounded matroid problem by Király, Lau and Singh [5] to get an integral solution.

Keywords: sparse graphs, forest covering, arboricity, matroid.

1 Introduction

Let $G = (V, E)$ be an undirected graph without loops. The set of edges induced by a node set $X \subseteq V$ is denoted by $E[X]$. The *arboricity* of G is defined as

$$\max_{X \subseteq V, |X| \geq 2} \frac{|E[X]|}{|X| - 1} .$$

A well-known result of Nash-Williams [8] states that a graph G can be covered by k forests if and only if its arboricity is at most k. If G has arboricity $k + \epsilon$ for some $0 < \epsilon < 1$, then this implies that it can be covered by $k+1$ forests, but not by k forests. It is natural to ask whether, if ϵ is small, then G can "almost" be covered by k forests in some sense. Recently, Montassier et al. [6] proposed a conjecture of that flavor, where "almost" means that the remaining edges form a forest of low maximum degree.

Conjecture 1 ([6]). *If the arboricity of G is at most $k + \epsilon$ for some $0 < \epsilon < 1$, then G decomposes into $k+1$ forests, one of which has maximum degree at most $\left\lceil \frac{(k+1)\epsilon}{1-\epsilon} \right\rceil$.*

⋆ Supported by grants OTKA CK80124 and TÁMOP 4.2.1./B-09/KMR-2010-0003. This work was done while the author visited The Chinese University of Hong Kong.
⋆⋆ Research supported by GRF grant 413609 from the Research Grant Council of Hong Kong.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 315–323, 2011.

This conjecture is best possible as shown by examples in [6]. Partial results are obtained by combinatorial method [6,4] and by topological method [3], and related results are known for planar graphs [1,2]. In this paper we are interested in a weaker form of the conjecture, where the bounded degree subgraph is not required to be a forest.

Conjecture 2. *If the arboricity of G is at most $k+\epsilon$ for some $0 < \epsilon < 1$, then G contains k forests such that the edges not covered by any of them form a subgraph of maximum degree at most $\left\lceil \frac{(k+1)\epsilon}{1-\epsilon} \right\rceil$.*

This weaker conjecture is also of interest by itself, and it has applications in bounding the game chromatic number [7]. Partial results towards this weaker conjecture are obtained in [7,1,6]. Recently, for $\epsilon \geq \frac{1}{2}$, Conjecture 2 was shown to be true by Kim et al. [4], but the case $\epsilon < \frac{1}{2}$ remains open (there are some special values for which it is known, see [4]). Our main result is the following theorem which almost proves Conjecture 2.

Theorem 3. *Let G be a graph with arboricity at most $k+\epsilon$, where k is a positive integer and $0 < \epsilon \leq \frac{1}{2}$. Then G contains k forests such that the edges not covered by any of them form a subgraph of maximum degree at most $\left\lceil \frac{(k+1)\epsilon}{1-\epsilon} \right\rceil + 1 = \left\lceil \frac{k\epsilon+1}{1-\epsilon} \right\rceil$.*

Unlike previous approaches, we use a linear programming based approach to tackle this problem. We first prove a fractional version of Conjecture 2 (see Theorem 4), and then show that Theorem 3 follows from a result of the degree bounded matroid problem [5]. A consequence of this approach is that the forests satisfying Theorem 3 can be constructed in polynomial time.

2 Relation to Degree Bounded Matroids

In the degree lower-bounded matroid independent set problem, we are given a matroid $M = (V, \mathcal{I})$, a hypergraph $H = (V, E)$, and lower bounds $f(e)$ for each hyperedge $e \in E(H)$. The task is to find an independent set I with $|I \cap e| \geq f(e)$ for each hyperedge $e \in E(H)$. The forest covering problem can be reduced to a degree lower-bounded independent set problem: It is a well-known consequence of the matroid union theorem that for any graph G and positive integer k there is a matroid M_k with ground set E whose independent sets are the edge sets that can be covered by k forests. Given an undirected graph $G = (V, E)$ and the forest covering problem with parameter k and Δ where Δ is the target maximum degree of the remaining graph, we set the matroid to be M_k and define the hypergraph H with $V(H) = E(G)$ and $E(H) = \{\delta(v) : v \in V(G)\}$ where $\delta(v)$ is the set of edges with exactly one endpoint in v, and set the lower bound for each hyperedge to be $d_G(v) - \Delta$ where $d_G(v) = |\delta(v)|$ is the degree of v in G. Then it can be seen that the degree bounded matroid problem in this setting is equivalent to the forest covering problem.

The result in [5] states that if there is a feasible solution to a linear programming relaxation of the degree bounded matroid problem, then there is an integral solution to the problem which violates the degree constraints by at most one[1]. The corresponding linear programming relaxation for the forest covering problem with parameter k is the following, where the objective is to minimize the maximum degree of the remaining graph. In the following let $d(v)$ denote the degree of node v in G, and for $x \in \mathbb{R}^E$ let $d_x(v) = \sum_{uv \in E} x_{uv}$.

$$\begin{aligned} \min \quad & \Delta && && (1)\\ \text{s.t.} \quad & x(E[X]) \le k(|X|-1) && \text{for every } \emptyset \neq X \subseteq V && (2)\\ & 0 \le x_e \le 1 && \text{for every } e \in E && (3)\\ & d_x(v) \ge d_G(v) - \Delta && \text{for every } v \in V && (4) \end{aligned}$$

The associated matroid polyhedron of M_k is described by (2) and (3). The requirement that $d_G(v) - d_x(v) \le \frac{(k+1)\epsilon}{1-\epsilon}$ for every $v \in V$ can be written as a degree lower bound for x by setting $\Delta = \frac{(k+1)\epsilon}{1-\epsilon}$:

$$d_x(v) \ge d_G(v) - \frac{(k+1)\epsilon}{1-\epsilon} \qquad \text{for every } v \in V. \tag{5}$$

The result in [5] states that if the system (2),(3),(5) has a solution, then the matroid has an independent set F which almost satisfies the degree bounds:

$$d_F(v) \ge d_G(v) - \left\lceil \frac{(k+1)\epsilon}{1-\epsilon} \right\rceil - 1 \qquad \text{for every } v \in V. \tag{6}$$

This would imply Theorem 3 if the system (2),(3),(5) was always feasible when the graph has arboricity at most $k+\epsilon$. We prove that this fractional version of Conjecture 2 is true.

Theorem 4. *Let G be a graph with arboricity at most $k+\epsilon$, where k is a positive integer and $0 < \epsilon \le \frac{1}{2}$. Then the system (2),(3),(5) has a feasible solution.*

We remark that this fractional version is also true if $\frac{1}{2} \le \epsilon < 1$, and it is in fact easier to prove. However, this is less interesting because in this case Conjecture 2 itself has been proved in [4], so we only sketch the proof at the end of Section 3.

An additional consequence of the method in [5] is that if we are given a cost function $c : E \to \mathbb{R}_+$, and the minimum cost of a solution of (2),(3),(5) is z_{LP}, then there are k forests with total cost at most z_{LP} that satisfy the condition of Theorem 3, and these can be found in polynomial time.

[1] More precisely the result in [5] applies to the degree bounded matroid basis problem where the returned solution is required to be a basis of the matroid, but it is easy to reduce the degree bounded matroid independent set problem to that problem by adding dummy variables and we omit the details here.

3 Proof of the Fractional Conjecture

Instead of trying to describe an optimal solution to the linear program described by (2),(3),(5), we will give an upper bound for the objective value of the dual linear program of (1)-(4) (when the arboricity of the graph is at most $k+\epsilon$), which is the following.

$$\begin{aligned}
\max \quad & \sum_{v\in V} d_G(v)\pi_v - \sum_{\emptyset\neq X\subseteq V} k(|X|-1)\mu_X - \sum_{e\in E}\rho_e \\
\text{s.t.} \quad & \pi_u + \pi_v - \sum_{Z:uv\in E[Z]} \mu_Z - \rho_{uv} \leq 0 \qquad \text{for every } uv\in E \\
& \sum_{v\in V}\pi_v \leq 1 \\
& \pi \geq 0 \\
& \mu \geq 0 \\
& \rho \geq 0
\end{aligned}$$

In an optimal dual solution $\rho_{uv} = \max\{\pi_u+\pi_v-\sum_{Z:uv\in E[Z]}\mu_Z, 0\}$. By writing $\sum_{v\in V} d_G(v)\pi_v = \sum_{uv\in E}(\pi_u+\pi_v)$ and eliminating the variables ρ, we get a simpler equivalent form.

$$\max \quad \sum_{uv\in E} \min\left\{\pi_u+\pi_v, \sum_{Z:uv\in E[Z]}\mu_Z\right\} - \sum_{\emptyset\neq X\subseteq V} k(|X|-1)\mu_X \tag{7}$$

$$\text{s.t.} \quad \sum_{v\in V}\pi_v \leq 1 \tag{8}$$

$$\pi \geq 0 \tag{9}$$

$$\mu \geq 0 \tag{10}$$

Let (π,μ) be an optimal dual solution. By duality, the following is equivalent to Theorem 4.

Theorem 5. *Let G be a graph with arboricity at most $k+\epsilon$, where k is a positive integer and $0<\epsilon\leq\frac{1}{2}$. Then*

$$\sum_{uv\in E} \min\left\{\pi_u+\pi_v, \sum_{Z:uv\in E[Z]}\mu_Z\right\} - \sum_{\emptyset\neq X\subseteq V} k(|X|-1)\mu_X \leq \frac{(k+1)\epsilon}{1-\epsilon}. \tag{11}$$

We will prove Theorem 5 in the rest of this section. Let $\mathcal{L} = \{\emptyset\neq X\subseteq V : \mu_X > 0\}$. By a standard uncrossing technique, we can simplify the optimal solution (π,μ) so that $\mathcal{L}$ is laminar, i.e. if X and Y in $\mathcal{L}$ are not disjoint, then $X\subseteq Y$ or $Y\subseteq X$.

Proposition 1. *We may assume that $\mathcal{L}$ is laminar.*

Proof. Suppose that X and Y in $\mathcal{L}$ are not disjoint. It is easy to verify that if we decrease μ_X and μ_Y by $\min\{\mu_X, \mu_Y\}$, and increase $\mu_{X\cap Y}$ and $\mu_{X\cup Y}$ by $\min\{\mu_X, \mu_Y\}$, then we obtain a feasible dual solution whose objective value is at least as large, since $\min\{\pi_u + \pi_v, \sum_{Z:uv\in E[Z]} \mu_Z\}$ would not decrease for any $uv \in E$ and the second summation remains the same. □

The overall plan of the proof is as follows. We give an upper bound for the first term on the left hand side of (11) in form of definite integrals in Proposition 2, and give lower bounds of the same form for the second term on the left hand side and also for the right hand side of (11). We then show in Lemma 1 that the required inequality holds for the integrands for any value of the variable, by using the assumption that the graph is of arboricity at most $k + \epsilon$.

Let us introduce some notation. For $X \in \mathcal{L}$, let $\alpha_X = \sum_{Z\supseteq X} \mu_Z$. Let $\alpha = \max\{\alpha_X : X \in \mathcal{L}\}$. For any $0 \le t \le \alpha$, let

$$\mathcal{L}_t = \{X \in \mathcal{L} : \alpha_X \ge t,\ \alpha_Y < t\ \forall Y \supsetneq X\}\ .$$

Note that the sets in $\mathcal{L}_t$ are disjoint because $\mathcal{L}$ is laminar. For any $0 \le t \le \alpha$ and $X \in \mathcal{L}_t$, let $X_t = \{v \in X :\ \pi_v \ge t\}$. Finally, given two node sets X and Y, let $d(X, Y)$ denote the number of edges with at least one endnode in both X and Y.

The first step of the proof is to give an upper bound for the first term of (11) that will turn out to be easier to estimate.

Proposition 2

$$\sum_{uv\in E} \min\left\{\pi_u + \pi_v, \sum_{Z:uv\in E[Z]} \mu_Z\right\} \le \int_0^\alpha \sum_{X\in\mathcal{L}_t} \left(\frac{1}{1-\epsilon}|E[X_t]| + d(X_t, X\setminus X_t) + \frac{1-2\epsilon}{1-\epsilon}|E[X\setminus X_t]|\right) dt\ .$$

Proof. The integral on the right hand side is in fact a finite sum, so it is well-defined. To prove the inequality, we show that the contribution of each edge to the right hand side is at least its contribution to the left side. Let $e = uv \in E$ be an arbitrary edge, and let us assume $\pi_u \ge \pi_v$. Let X be the smallest set in $\mathcal{L}$ that contains both u and v; thus $\sum_{Z:uv\in E[Z]} \mu_Z = \alpha_X$. For any $t \in [0, \alpha_X]$, there is exactly one set $Z \in \mathcal{L}_t$ with $u, v \in Z$ since $\mathcal{L}$ is laminar and thus the sets in $\mathcal{L}_t$ are disjoint. We distinguish three cases.

1. $\pi_u \ge \alpha_X$. In this case the contribution of e to the left hand side is equal to α_X, and we will show that its contribution to the right hand side is at least α_X. When $t \in [0, \min\{\alpha_X, \pi_v\}]$, edge e is counted with weight $\frac{1}{1-\epsilon}$ in the right hand side because both u and v are in Z_t. If $\pi_v \ge \alpha_X$ then we are done. Otherwise e is counted with weight 1 when $t \in [\pi_v, \alpha_X]$ because $u \in Z_t$ but $v \notin Z_t$. Therefore the total contribution of e is at least α_X.

2. $\pi_u < \alpha_X \leq \pi_u + \pi_v$. In this case the contribution of e to the left hand side is equal to α_X. In the right hand side, the edge e is counted with weight $\frac{1}{1-\epsilon}$ if $t \in [0, \pi_v]$ when both $u, v \in Z_t$, with weight 1 if $t \in [\pi_v, \pi_u]$ when $u \in Z_t$ and $v \notin Z_t$, and with weight $\frac{1-2\epsilon}{1-\epsilon}$ if $t \in [\pi_u, \alpha_X]$ when both $u, v \notin Z_t$. Thus the total contribution of e to the right hand side is equal to

$$\frac{1}{1-\epsilon}\pi_v + (\pi_u - \pi_v) + \frac{1-2\epsilon}{1-\epsilon}(\alpha_X - \pi_u) = \frac{1-2\epsilon}{1-\epsilon}\alpha_X + \frac{\epsilon}{1-\epsilon}\pi_u + \frac{\epsilon}{1-\epsilon}\pi_v \ .$$

   Since $\pi_u + \pi_v \geq \alpha_X$ by assumption, this is at least α_X as desired.
3. $\pi_u + \pi_v \leq \alpha_X$. In this case the contribution of e to the left hand side is equal to $\pi_u + \pi_v$. The contribution of e to the right hand side is the same as above: $\frac{1}{1-\epsilon}$ if $t \in [0, \pi_v]$, 1 if $t \in [\pi_v, \pi_u]$, and $\frac{1-2\epsilon}{1-\epsilon}$ if $t \in [\pi_u, \alpha_X]$, and thus the total contribution is equal to

$$\frac{1}{1-\epsilon}\pi_v + (\pi_u - \pi_v) + \frac{1-2\epsilon}{1-\epsilon}(\alpha_X - \pi_u) \ .$$

   Since $\alpha_X - \pi_u \geq \pi_v$ and $\frac{1}{1-\epsilon} + \frac{1-2\epsilon}{1-\epsilon} = 2$, the contribution of e to the right hand side is at least $2\pi_v + (\pi_u - \pi_v) = \pi_u + \pi_v$ as desired (note that here we use the assumption that $\epsilon \leq \frac{1}{2}$). □

We reformulate the second term on the left side of (11) as an integral on the interval $[0, \alpha]$:

$$\sum_{X \subseteq V} k(|X| - 1)\mu_X = \int_0^\alpha \sum_{X \in \mathcal{L}_t} k(|X| - 1)\, dt \ .$$

The next step is to lower bound the constant on the right hand side of (11) by an integral with the same limits. Let us use the notation $\pi(X) = \sum_{v \in X} \pi_v$. By (8) we have

$$1 \geq \pi(V) \geq \sum_{v \in V} \min\{\pi_v, \sum_{Z : v \in Z} \mu_Z\} = \int_0^\alpha \sum_{X \in \mathcal{L}_t} |X_t|\, dt \ ,$$

where the equality follows because the contribution of v to the right hand side is equal to $\min\{\pi_v, \sum_{Z : v \in Z} \mu_Z\}$. Thus

$$\frac{(k+1)\epsilon}{1-\epsilon} \geq \int_0^\alpha \sum_{X \in \mathcal{L}_t} \frac{(k+1)\epsilon}{1-\epsilon} |X_t|\, dt \ .$$

After these formulations, to prove Theorem 5, it suffices to show that

$$\int_0^\alpha \sum_{X \in \mathcal{L}_t} \left(\frac{1}{1-\epsilon} |E[X_t]| + d(X_t, X \setminus X_t) + \frac{1-2\epsilon}{1-\epsilon} |E[X \setminus X_t]| \right) dt$$
$$\leq \int_0^\alpha \sum_{X \in \mathcal{L}_t} \left(\frac{(k+1)\epsilon}{1-\epsilon} |X_t| + k(|X| - 1) \right) dt \ . \quad (12)$$

We show that the inequality holds for the integrands for any value of t between 0 and α, so it holds for the integrals as well. The assumption that G is of arboricity at most $k+\epsilon$ is only used in the following lemma.

Lemma 1. *For any $0 \le t \le \alpha$ and $X \in \mathcal{L}_t$, the following inequality holds:*

$$\frac{1}{1-\epsilon}|E[X_t]| + d(X_t, X \setminus X_t) + \frac{1-2\epsilon}{1-\epsilon}|E[X \setminus X_t]| \le k(|X|-1) + \frac{(k+1)\epsilon}{1-\epsilon}|X_t| \ .$$

Proof. The idea is to identify the high degree structures Y in $X \setminus X_t$, and then use the arboricity to bound $|E(X_t \cup Y)|$, while the number of remaining edges can be bounded by $k|X \setminus (Y \cup X_t)|$. Let $C_1, \ldots, C_l$ be the components of $G[X \setminus X_t]$, and let Y be the union of the components where the average degree in G of the nodes is at least $k+1$, i.e.

$$Y = \bigcup \{C_i : 2|E[C_i]| + d(C_i, X_t) \ge (k+1)|C_i|\} \ .$$

Proposition 3. *The following two inequalities hold for this set Y:*

$$2|E[Y]| + d(Y, X_t) \ge (k+1)|Y| \ , \tag{13}$$

$$d(X \setminus (Y \cup X_t), X) \le k|X \setminus (Y \cup X_t)| \ . \tag{14}$$

Proof. Inequality (13) follows easily from the definition, since it holds for all components of $G[Y]$. To show inequality (14), observe that if $C_i \cap Y = \emptyset$, then $2|E[C_i]| + d(C_i, X_t) \le k|C_i| + (|C_i|-1)$. This implies, using that $|E[C_i]| \ge |C_i|-1$ because of its connectedness, that $|E[C_i]| + d(C_i, X_t) \le k|C_i|$. By summing over all components not in Y, we obtain that

$$d(X \setminus (Y \cup X_t), X) = \sum_{i : C_i \cap Y = \emptyset} (|E[C_i]| + d(C_i, X_t)) \le k|X \setminus (Y \cup X_t)| \ . \qquad \square$$

First let us analyze the case when $X_t \cup Y = \emptyset$. Since all components have average degree less than $k+1$, we have $|E[X]| \le \frac{k+1}{2}|X| - \frac{1}{2}$. A simple case analysis shows (using the fact that G has no loops) that this implies $|E[X]| \le k(|X|-1)$, so the Lemma is true in this case.

We may thus assume that $X_t \cup Y \neq \emptyset$. Since the arboricity of G is at most $k+\epsilon$, we know that $|E[X_t \cup Y]| \le (k+\epsilon)(|X_t \cup Y| - 1)$, so

$$\frac{1}{1-\epsilon}(|E[X_t]| + d(X_t, Y) + E[Y]) = \frac{1}{1-\epsilon}|E[X_t \cup Y]| \le \frac{k+\epsilon}{1-\epsilon}(|X_t \cup Y| - 1) \ .$$

If we subtract $\frac{\epsilon}{1-\epsilon}$ times the inequality (13) from this, we get that

$$\begin{aligned}
&\frac{1}{1-\epsilon}|E[X_t]| + d(X_t, Y) + \frac{1-2\epsilon}{1-\epsilon}|E[Y]| \\
&\le \frac{k+\epsilon}{1-\epsilon}(|X_t \cup Y| - 1) - \frac{(k+1)\epsilon}{1-\epsilon}|Y| \\
&= \left(k + \frac{(k+1)\epsilon}{1-\epsilon}\right)(|X_t \cup Y| - 1) - \frac{(k+1)\epsilon}{1-\epsilon}|Y| \\
&= k(|X_t \cup Y| - 1) + \frac{(k+1)\epsilon}{1-\epsilon}(|X_t| - 1) \ .
\end{aligned}$$

Next we add inequality (14):

$$\begin{aligned}
&\frac{1}{1-\epsilon}|E[X_t]| + d(X_t, X \setminus X_t) + \frac{1-2\epsilon}{1-\epsilon}|E[Y]| + E[X \setminus (Y \cup X_t)] \\
&\leq k(|X_t \cup Y| - 1) + \frac{(k+1)\epsilon}{1-\epsilon}(|X_t| - 1) + k|X \setminus (Y \cup X_t)| \\
&= k(|X| - 1) + \frac{(k+1)\epsilon}{1-\epsilon}(|X_t| - 1) \ .
\end{aligned}$$

This implies the inequality in the Lemma because

$$\frac{1-2\epsilon}{1-\epsilon}|E[Y]| + E[X \setminus (Y \cup X_t)] \geq \frac{1-2\epsilon}{1-\epsilon}|E[X \setminus X_t]| \ . \qquad \square$$

By Lemma 1, inequality (12) is true, since the inequality holds for the integrands for any value $t \in [0, \alpha]$. This concludes the proof of Theorem 5, hence also the proof of Theorem 4. Using the degree bounded matroid result described in Section 2, we obtain Theorem 3.

Remark. As we have already mentioned, Theorems 4 and 5 are true also for $\frac{1}{2} \leq \epsilon < 1$. We now sketch the proof. The overall structure of the proof is similar, but we remove the term $\frac{1-2\epsilon}{1-\epsilon}|E[X \setminus X_t]|$ from the bound in Proposition 2. Therefore Lemma 1 should be modified: the inequality

$$\frac{1}{1-\epsilon}|E[X_t]| + d(X_t, X \setminus X_t) \leq k(|X| - 1) + \frac{(k+1)\epsilon}{1-\epsilon}|X_t|$$

should hold for any $0 \leq t \leq \alpha$ and $X \in \mathcal{L}_t$. The proof of this is simpler than the proof of Lemma 1: instead of considering the components of $X \setminus X_t$, we define Y as the set of nodes of $X \setminus X_t$ for which $d(v, X_t) \geq k+1$. Using the fact that $|E[X_t \cup Y]| \leq (k+\epsilon)(|X_t \cup Y| - 1)$ and the fact that $d(v, X_t) \leq k$ for any $v \notin X_t \cup Y$, we obtain the desired bound.

Acknowledgement

We thank Hehui Wu for telling us this problem and their result [4].

References

1. Borodin, O.V., Kostochka, A.V., Sheikh, N.N., Yu, G.: Decomposing a planar graph with girth 9 into a forest and a matching. European J. Combin. 29, 1235–1241 (2008)
2. Goncalves, D.: Covering planar graphs with forests, one having bounded maximum degree. J. Combin. Theory Ser. B 99, 314–322 (2009)
3. Kaiser, T., Montassier, M., Raspaud, A.: Covering a graph by forests and a matching. arXiv:1007.0316v1 (2010)
4. Kim, S.-J., Kostochka, A.V., West, D.B., Wu, H., Zhu, X.: Decomposition of sparse graphs into forests and a graph with bounded degree (2010) (submitted)

5. Király, T., Lau, L.C., Singh, M.: Degree bounded matroids and submodular flows. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO 2008. LNCS, vol. 5035, pp. 259–272. Springer, Heidelberg (2008)
6. Montassier, M., Ossona de Mendez, P., Raspaud, A., Zhu, X.: Decomposing a graph into forests (2010) (manuscript)
7. Montassier, M., Pecher, A., Raspaud, A., West, D.B., Zhu, X.: Decomposition of sparse graphs, with application to game coloring number. Discrete Mathematics 310, 1520–1523 (2010)
8. Nash-Williams, C.S.J.A.: Decomposition of finite graphs into forests. J. London Math. Soc. 39(12) (1964)

A Primal-Dual Algorithm for Weighted Abstract Cut Packing

S. Thomas McCormick[1,⋆] and Britta Peis[2,⋆⋆]

[1] University of British Columbia, Sauder School of Business, 2053 Main Mall, Vancouver, BC V6T1Z2, Canada
tom.mccormick@sauder.ubc.ca
[2] Technische Universität Berlin, Strasse des 17. Juni 136, 10623 Berlin, Germany
peis@math.TU-Berlin.de

Abstract. Hoffman and Schwartz [13] developed the Lattice Polyhedron model and proved that it is totally dual integral (TDI), and so has integral optimal solutions. The model generalizes many important combinatorial optimization problems such as polymatroid intersection, cut covering polyhedra, min cost arborescences, etc., but has lacked a combinatorial algorithm. The problem can be seen as the blocking dual of Hoffman's Weighted Abstract Flow (WAF) model [11], or as an abstraction of ordinary Shortest Path and its cut packing dual, so we call it Weighted Abstract Cut Packing (WACP). We develop the first combinatorial algorithm for WACP, based on the Primal-Dual Algorithm framework. The framework is similar to that used in [14] for WAF, in that both algorithms depend on a relaxation by a scalar parameter, and then need to solve an unweighted "restricted" subproblem. The subroutine to solve WACP's restricted subproblem is quite different from the corresponding WAF subroutine. The WACP subroutine uses an oracle to solve a restricted abstract cut packing/shortest path subproblem using greedy cut packing, breadth-first search, and an update that achieves complementary slackness. This plus a standard scaling technique yields a polynomial combinatorial algorithm.

1 Introduction

1.1 Overview

A fundamental problem in polyhedral combinatorics is deciding which linear programs have guaranteed integer optimal solutions. The canonical example here is the Max Flow/Min Cut Theorem [4], where integrality stems from the matrices being totally unimodular (TUM). But TUM is too strong in the sense that essentially only network flow LPs are TUM [20].

A more general tool for this has been the concept of *total dual integrality (TDI)* [18], where LPs with specially structured RHSs (often submodular) can be proven to have integral optimal solutions. Development of polynomial algorithms to solve TDI

⋆ Supported by an NSERC Operating Grant.
⋆⋆ Supported by a DFG grant.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 324–335, 2011.

problems has lagged behind the theoretical results, although there have been some recent successes: Submodular Function Minimization [16] and Bisubmodular Function Minimization [17].

A key example of a TDI but not TUM class of LPs is the classic "path packing" version of Max Flow/Min Cut. Hoffman [11] found a way to get an abstract version of this, and an algorithm for solving *Weighted Abstract Flow (WAF)* was finally developed in [14].

WAF abstracts "paths" as subsets of elements (e.g., arcs). It is well-known that *s–t* cuts can be characterized as the minimal subsets of elements that hit (have non-empty intersection with) every *s–t* path (and vice versa), so that minimal paths and cuts are *blocking clutters* [8]. In [12] Hoffman worked out that the blocking clutter of abstract paths are an abstraction of cuts that arise in *Lattice Polyhedra* [13].

The blocking dual of the classic path packing version of Max Flow is a cut packing LP that is well-understood to be the dual LP to a formulation of Dijkstra Shortest Path. In this sense we can view Lattice Polyhedra as generalizing Shortest Path in the same sense that WAF generalizes Max Flow. We call this problem *Weighted Abstract Cut Packing (WACP)*.

Thus it is natural to ask for an algorithm for WACP to go with the [14] algorithm for WAF. The main result in this paper is such an algorithm. Like the WAF algorithm, it is based on the Primal-Dual framework. This framework requires a subalgorithm for solving the corresponding unweighted problem. For WAF such a subalgorithm was given by [15]; for WACP, Frank [5] gives such an algorithm. In each case we need to generalize the unweighted algorithm to solve a so-called *restricted* problem, which is non-trivial. These algorithms can be seen as special cases of the framework developed in [1]. This extended abstract omits most proofs and figures.

1.2 The WACP Model

Given a binary matrix $B \in \{0,1\}^{|\mathscr{L}|\times|E|}$, a "cost function" $c \in \mathbb{Z}_+^{|E|}$ and a "rank function" $r \in \mathbb{Z}_+^{|\mathscr{L}|}$, define $\mathbb{P}(B,r) := \{x \in \mathbb{R}^{|E|} \mid Bx \geq r,\ x \geq 0\}$. We are interested in finding optimal (integral) solutions of the primal-dual pair of linear programs

$$\text{(P)} \quad \min\{c^T x \mid x \in \mathbb{P}(B,r)\}, \quad \text{and} \quad \text{(D)} \quad \max\{r^T y \mid B^T y \leq c,\ y \geq 0\}.$$

Such covering and packing problems are NP-hard in general. However, if $\mathbb{P}(B,r)$ is a *lattice polyhedron* (defined below), Hoffman and Schwartz [13] prove that the underlying system of linear inequalities is TDI, implying that $\mathbb{P}(B,r)$ has only integral vertices. Each row $i \in \mathscr{L}$ is the incidence vector of a subset $S_i \subseteq 2^E$, and so we identify $\mathscr{L}$ with a family of subsets $\mathscr{L} \subseteq 2^E$.

Definition 1 (Lattice polyhedron [13]). $\mathbb{P}(B,r)$ *is a* lattice polyhedron *if the row set* $\mathscr{L}$ *forms a lattice* $\mathscr{L} = (\mathscr{L}, \preceq, \wedge, \vee)$ *such that r is supermodular on* $\mathscr{L}$*, and* $\mathscr{L}$ *satisfies*

(C1) $i \prec j \prec k$ *implies* $S_i \cap S_k \subseteq S_j$ *for all* $i,j,k \in \mathscr{L}$ *(then we call* $(\mathscr{L}, \preceq)$ consecutive*), and*

(C2) $(S_i \vee S_j) \cup (S_i \wedge S_j) \subseteq S_i \cup S_j$ *for all* $i,j \in \mathscr{L}$.

Note that a consecutive lattice is submodular if and only if each column of B is submodular, i.e., iff for each non-negative $x \in \mathbb{R}_+^{|E|}$ the extension of x to $\mathscr{L}$ via $x(S) := \sum_{e \in S} x_e$ satisfies

$$(Sub) \quad x(S) + x(T) \geq x(S \vee T) + x(S \wedge T) \quad \forall S, T \in \mathscr{L}.$$

We say that a consecutive lattice is *submodular* if (Sub) is satisfied, and *modular* if (Sub) is satisfied with equality.

Note that a lattice polyhedron generalizes a(n anti-)polymatroid (given by a supermodular rank function defined on the Boolean lattice $(2^E, \subseteq, \cup, \cap)$), since the Boolean lattice is easily seen to be consecutive and modular. Beside polymatroids, lattice polyhedra also cover various other classical discrete structures such as the intersection of polymatroids, cut covering polyhedra, min cost aborescences, etc. (see [18]).

Example 1. Let $G = (V, E)$ be a (directed or undirected) graph with source s and sink t and consider the collection $\mathscr{L}(G) = \{\delta(S) \subseteq E \mid s \in S \subseteq V,\ t \notin S\}$ of all s–t cuts in G (as usual, $\delta(S) := \{(u, v) \in E \mid u \in S,\ v \notin S\}$). Note that $\mathscr{L}(G)$ forms a consecutive submodular lattice with respect to the ordering $\delta(S) \preceq \delta(T) \iff S \subseteq T \quad \forall S, T \subseteq V$. Thus, with constant rank function $r \equiv 1$ the LPs (P) and (D) are the ordinary shortest s–t path and its dual cut packing problems.

Thus we call the primal linear program (P) an *abstract shortest path problem*, and its dual (D) a *weighted abstract cut packing problem (WACP)*. Even though efficient algorithms exist for all of the examples of WACP mentioned above, [13]'s TDI result does not have an algorithm. The most general algorithm to this point is Frank's two-phase greedy algorithm [5] for lattice polyhedra with monotone rank function, which requires that r satisfies the monotonicity property $D \preceq Q \Longrightarrow r_D \leq r_Q \quad \forall D, Q \in \mathscr{L}$. Note that this property is satisfied for the unweighted case where $r \equiv 1$, but not for the polymatroid intersection case. Here we develop an algorithm for lattice polyhedra in general, i.e., where r is not necessarily monotone.

2 Preliminaries

We call $D \in \mathscr{L}$ a *cut*, and any $y \in \mathbb{R}^{\mathscr{L}}$ a *cut packing*. Each cut packing induces a mass on each element, denoted by $y(e) := \sum_{D \ni e} y_D$. Thus y is feasible iff $y \geq 0$ and $y(e) \leq c_e$ for all $e \in E$. If $y(e) = c_e$ we say that e is *tight* (w.r.t. y). Any $x \in \mathbb{R}^E$ is a *primal vector*, which induces a mass on each D via $x(D) = \sum_{e \in D} x_e$. Then x is feasible iff $x \geq 0$ and $x(D) \geq r_D$ for all $D \in \mathscr{L}$. Suppose that y is a feasible cut packing. Then $x \geq 0$ and y are jointly optimal iff (OPT i) $x(D) \geq r_D$ for all $D \in \mathscr{L}$; (OPT ii) $y_D \cdot (x(D) - r_D) = 0$ for all $D \in \mathscr{L}$; and (OPT iii) $x_e \cdot (c_e - y(e)) = 0$ for all $e \in E$. Condition (OPT i) is just primal feasibility, and (OPT ii–iii) are complementary slackness.

2.1 Accessing the Abstract Network via an Oracle

We take the point of view that $|E|$ is "small", and so we explicitly store x as a vector in $\mathbb{R}^E$. Denote $m = |E|$. Let h denote an upper bound on the height of $\mathscr{L}$, so that any chain in $\mathscr{L}$ has at most h cuts. We think of $|\mathscr{L}|$ as being "big", possibly exponential in m, so we do not have an explicit list of cuts and their r-values. We represent WACP with an

oracle for (P) that when given a vector x, either verifies that x is feasible, or returns a constraint violated by x. In Ellipsoid terms, it is a separation routine (see [10]). We use this oracle to generate interesting cuts in $\mathscr{L}$ as needed. We keep only a list of the current positive cuts and their respective y values. A "small" E and "big" $\mathscr{L}$ is consistent with most applications of this model. Formally the oracle is:

$\mathscr{O}(x)$ when given $x \in \mathbb{R}_+^E$, returns either a *violating cut* D with $x(D) < r_D$ (together with r_D), or the statement that every $D \in \mathscr{L}$ satisfies primal feasibility, i.e., (OPT i).

Let CO denote the time for one call to $\mathscr{O}(x)$. The results in [10] imply that $\mathscr{O}(x)$ is enough to get a strongly polynomial algorithm for WACP. However, Ellipsoid cannot produce *integral* optimal solutions to the dual in TDI problems, i.e., integral optimal cut packing y for (D). Let C be an upper bound on the c_e and set $r_{\max} = \max_{D \in \mathscr{L}} r_D$ so the size of the data is $\log C + \log r_{\max}$. A pseudopolynomial bound can involve C, $r_{\max}$, m, h, and CO, a weakly polynomial bound can involve $\log C$, $\log r_{\max}$, m, h, and CO, and a strongly polynomial bound can involve only m, h, and CO.

3 Outer Framework of the Algorithm

The outer framework of our algorithm is similar to the WAF algorithm in [14]. Both algorithms can be viewed as special cases of the Primal-Dual Algorithm (a.k.a. Successive Shortest Path). In both cases we relax the "supermodular constraints" with RHS r by relaxing r to $r - \lambda$ for scalar parameter λ. However, since WACP is the blocking dual of WAF, it interchanges the roles of primal and dual, so the relaxed r constraint is in the dual of WAF, but in the primal here.

The relaxed constraint is $x(D) \geq r_D - \lambda$; defining $\mathrm{gap}_\lambda(D) = x(D) - r_D + \lambda$ (usually just $\mathrm{gap}(D)$) this is $\mathrm{gap}(D) \geq 0$. The relaxed problem replaces (OPT i–iii) by (OPT(λ) i) $\mathrm{gap}(D) \geq 0$; (OPT(λ) ii) $y_D \cdot \mathrm{gap}(D) = 0$; and (OPT($\lambda$) iii) $x_e \cdot (c_e - y(e)) = 0$. Define the sublattice (or abstract sub-network) $\mathscr{L}(\lambda) := \{D \in \mathscr{L} \mid \mathrm{gap}(D) = 0\}$.

Lemma 1. *$\mathscr{L}(\lambda)$ is a consecutive modular sublattice of $\mathscr{L}$.*

Proof. (C1) is trivially satisfied as $\mathscr{L}(\lambda) \subseteq \mathscr{L}$. To prove that $\mathscr{L}(\lambda)$ satisfies (Sub) with equality, consider $D, T \in \mathscr{L}(\lambda)$. Then $\mathrm{gap}(D \vee T) + \mathrm{gap}(D \wedge T) \geq 0$ follows from primal feasibility. On the other hand, the submodularity of $\mathscr{L}$ and the supermodularity of $r - \lambda \mathbb{1}^T$ imply $\mathrm{gap}(D \vee T) + \mathrm{gap}(D \wedge T) \leq \mathrm{gap}(D) + \mathrm{gap}(T) = 0$.

Idea of the algorithm: At iteration i the algorithm computes optimal integral solutions $x^{(i)}$ and $y^{(i)}$ for the relaxed problems (P$^{(i)}$) and (D$^{(i)}$) which arise from (P) and (D) by replacing r by $r - \lambda_i$. The initial value of λ is $\lambda_0 = r_{\max}$. Notice that $x^{(0)} = y^{(0)} = 0$ are optimal for λ_0, and that if $\lambda_k = 0$, then $x^{(k)}$ and $y^{(k)}$ are optimal to (P) and (D). We will show that the λ_i's are positive integers satisfying $\lambda_i > \lambda_{i+1}$ for $i = 1, \ldots$, and so the algorithm terminates after at most $r_{\max}$ steps with optimal solutions. The algorithm also monotonically increases $\mathbb{1}^T y$, which is bounded by $O(mC)$. We can show that $O(m)$ iterations suffice to strictly increase $\mathbb{1}^T y$, and so $O(m^2 C)$ iterations overall. Then Section 3.2 and Theorem 5 imply the pseudo-polynomial bound:

Theorem 1. *The algorithm solves WACP in* $O(\min\{r_{\max}, m^2C\}\log r_{\max}(m+\mathrm{CO})(m\cdot \mathrm{CO}+h(m+h)))$ *time.*

3.1 Detailed Algorithm

Initialization: The algorithm starts with $\lambda_0 = r_{\max}$, and the optimal solutions $x^{(0)} = 0$ and $y^{(0)} = 0$ of ($\mathrm{P}^{(0)}$) and ($\mathrm{D}^{(0)}$), resp.

General step ($i \to i+1$)*:* The step begins with an optimal integral solution $x^{(i)}$ of ($\mathrm{P}^{(i)}$), an optimal integral solution $y^{(i)}$ of ($\mathrm{D}^{(i)}$) whose support forms a chain in $\mathscr{L}$, the "restricted" elements $R^{(i)} = \{e \in E \mid x_e^{(i)} > 0\}$, and the consecutive modular sublattice (see Lemma 1) $\mathscr{L}^{(i)} := \mathscr{L}(\lambda_i)$. Notice that (OPT($\lambda$) iii) requires that when $x_e > 0$, then $y(e) = c_e$. Thus we must find a cut packing y that preserves $y(e) = c_e$ on $R^{(i)}$. The algorithm calls a subroutine (see Section 4) to find an optimal solution $p^* \in \{-1,0,1\}^E$ of the *restricted abstract shortest path problem*

$$(RASP_i) \quad \min\{c^T p \mid p(D) \geq \mathbb{1}\ (D \in \mathscr{L}^{(i)}),\ p_e \geq 0\ (e \notin R^{(i)})\},$$

and an optimal solution $z^* \in \mathbb{Z}^{\mathscr{L}^{(i)}}$ of the *restricted abstract cut packing problem*

$$(RACP_i) \quad \max_{z \geq 0}\{\mathbb{1}^T z \mid z(e) \leq c_e\ (e \in E),\ z(e) = c_e\ (e \in R^{(i)})\},$$

whose support forms a chain in $\mathscr{L}$. Moreover, the algorithm computes an integral step length $0 < \theta_i \leq \lambda_i$.

Update: It then updates $y^{(i+1)} \leftarrow z^*$, $x^{(i+1)} \leftarrow x^{(i)} + \theta_i p^*$, and $\lambda_{i+1} \leftarrow \lambda_i - \theta_i$.

Note that RASP_i has RHS $\mathbb{1}$, and so RASP really is a type of Shortest Path problem that is being used to find an update direction p^* for x. RACP_i is an unweighted version of WACP being used to find the new y. As we will see in Section 4, our algorithm for RACP and RASP is quite different from the corresponding algorithm for WAF's restricted subproblems in [14]. But before we consider RACP and RASP, we show how to determine a suitable step length θ, and show that (provided that the restricted problems can be solved) at each iteration i the algorithm finds optimal integral primal and dual solutions for the relaxed problems ($\mathrm{P}^{(i)}$) and ($\mathrm{D}^{(i)}$).

3.2 The Choice of Step Length θ

Each iteration starts with a feasible integral solution x for a current integral λ, an integral RACP z^* whose support forms a chain, and an integral optimal RASP p^*. For simplicity, denote λ^i and x^i by λ and x, and λ^{i+1} and x^{i+1} by λ' and x'. Define $\mathrm{gap}'(D) = x'(D) - r_D + \lambda' = \mathrm{gap}(D) - \theta(1 - p^*(D))$. Then the algorithm chooses θ maximal such that (1)–(3) are satisfied:

(1) $\theta \leq \lambda$ (to ensure $\lambda' \geq 0$),
(2) $\theta \leq \min\{x_e \mid p^*(e) < 0\}$ (to ensure $x' \geq 0$), and
(3) $\theta \leq \min\{\frac{\mathrm{gap}(D)}{1-p^*(D)} \mid p^*(D) \leq 0\}$ (to ensure that $x'(D) \geq r_D - \lambda'$, i.e., $\mathrm{gap}'(D) \geq 0$).

Theorem 2. *This θ is a positive integer, and x' is an integral feasible solution.*

Since λ is an integer and $p^*(e) < 0$ only if $x_e > 0$, if θ is determined by (1) or (2) it is clearly an integer. Suppose instead that θ is determined by (3). We show that in this case there exists a minimizer D_0 of the constraint in (3) with $p^*(D_0) = 0$. Since $\mathrm{gap}(D) > 0$ for all $D \in \mathscr{L}$ with $p^*(D) \leq 0$, this would imply that $\theta = \mathrm{gap}(D_0)$ is a positive integer. The following lemmas prove that such a D_0 exists.

Lemma 2. *Suppose*

$$\theta = \min\{\frac{gap(D)}{1-p^*(D)} \mid p^*(D) \leq 0\} < \min\{x_e \mid p^*(e) < 0\}, \tag{1}$$

and let D, Q satisfy $gap'(D) = gap'(Q) = 0$. Then for $V \in \{D \vee Q, D \wedge Q\}$: (i) $gap'(V) = 0$; (ii) $p^*(D) + p^*(Q) = p^*(D \vee Q) + p^*(D \wedge Q)$; (iii) $\theta = \frac{gap(V)}{1-p^*(V)}$ if $p^*(V) \neq 1$; and (iv) $p^*(V) \leq 1$.

Proof. Note that the choice of θ implies $x' \geq 0$, and so $\mathrm{gap}' = x' - r + \lambda'$ is submodular. The choice of θ ensures that $\mathrm{gap}' \geq 0$, so that $0 = \mathrm{gap}'(D) + \mathrm{gap}'(Q) \geq \mathrm{gap}'(D \vee Q) + \mathrm{gap}'(D \wedge Q) \geq 0$, implying (i). However, this equality holds only if all $e \in \mathrm{supp}(x')$ are modular in the sublattice $\{D, Q, D \vee Q, D \wedge Q\}$. Since $\mathrm{supp}(p^*) \subseteq \mathrm{supp}(x')$ by our assumption, property (ii) follows. Finally, (iii) and (iv) follow from $\mathrm{gap}' = \mathrm{gap} - \theta(1 - p^*)$ and $\mathrm{gap} \geq 0$.

In order to prove that there exists a D_0 such that $\theta = \mathrm{gap}(D_0)$ whenever (3) determines θ, we consider a chain $\mathscr{C}' = \{Q_k \prec \cdots \prec Q_1 \prec Q_0\}$ in $\mathscr{L}(\lambda)$ with $p^*(Q_i) = 1$ for all $Q_i \in \mathscr{C}'$. We will see below how to choose such a chain. It will follow as a consequence of the subroutines described in Section 4 that (P1) $Q_k \cap \{e \in E \mid p^*(e) < 0\} = \emptyset$ and (P2) $p^*(D) \geq 0$ whenever $Q_i \preceq D \preceq Q_{i+1} (i = k, \ldots, 1)$. These properties are used in:

Lemma 3. *Let θ be as in Lemma 2 and let D_0 be a minimizer with largest value $p^*(D_0)$. Moreover, assume that D_0 is maximal w.r.t. $\preceq$ among all such lattice elements. Then $p^*(D_0) = 0$.*

Thus Lemma 3 proves Theorem 2. We use this to compute θ using binary search in $O(\log r_{\max}(m + \mathrm{CO}))$ time.

Corollary 1. *$y' = z^*$ and x' are integral optimal solutions of $(P(\lambda'))$ and $(D(\lambda'))$.*

Proof. The feasibility of x' follows from Theorem 2. To show that $y' = z^*$ is feasible, we need to show that $\sum_{D \in \mathscr{L}: e \in D} z^*_D \leq c_e \quad \forall e \in E$. However, this immediately follows from $\sum_{D \in \mathscr{L}: e \in D} z^*_D = \sum_{D \in \mathscr{L}(\lambda): e \in D} z^*_D$ (as $\mathrm{supp}(z^*) \subseteq \mathscr{L}(\lambda) \subseteq \mathscr{L}$).

It remains to show that z^* and x' satisfy the complementary slackness conditions (OPT(λ) ii–iii). To prove (OPT(λ) iii), observe that x'_e can only be positive if either $x_e > 0$ or $p^*_e > 0$. In the former, $e \in R$ implying $z^*(e) = c_e$. In the latter, $z^*(e) = c_e$ follows from the complementary slackness of z^* and p^*. To prove (OPT(λ) ii), note that $z^*_D > 0$ implies that $D \in \mathscr{L}(\lambda)$ and $p^*(D) = 1$ (again, by complementary slackness of z^* and p^*). It follows that $x'(D) = x(D) + \theta p^*(D) = r_D - \lambda + \theta = r_D - \lambda'$.

3.3 Polynomial Algorithm via Scaling

To make the algorithm polynomial, in theory we could choose to scale either the rewards r_D or the costs c_e. It is not clear how to scale the r_D so that supermodularity is preserved. Hence we scale the costs to get a (weakly) polynomial algorithm for WACP. Recall that C is the largest cost of any element, and set $b = \lfloor \log_2 C \rfloor = O(\log C)$. For $0 \le k \le b+1$, define $c_e^k = \lfloor c_e/2^k \rfloor$, the k-th *scale* of c. Thus $c^{b+1} = 0$, c^b is a 0–1 vector, and $c^0 = c$.

In Theorem 1, when $C = 1$ the algorithm is polynomial. Clearly $2y$ and x are optimal for capacities $2c^b$. Note that $2c^b$ is "close" to c^{b-1} in the sense that $c_e^{b-1} - 2c_e^b$ is 0 or 1 for all $e \in E$. Define the set of elements whose capacities need to be incremented at scale k as $I(k) = \{e \mid c_e^{k-1} - 2c_e^k = 1\}$, and $\chi(e) \in \mathbb{R}^E$ as the characteristic vector of e. We would like to replace $c' = c^b$ by $c' + \chi(e)$ for each $e \in I(b)$ and then modify the optimal x, y for c' to a new optimal x', y' for $c' + \chi(e)$. Notice that if $x_e = 0$, then $y' = y$ and $x' = x$ are again optimal for $c' + \chi(e)$, and so the algorithm first deletes all such elements from $I(b)$. The challenge is to modify x and y so they are optimal for r and $c + \chi(f)$, for $f \in I(k)$. If y changes it does so on a generalized type of cut containing f which is a symmetric difference of cuts.

To deal with this we re-use the techniques of the RASP/RACP subroutine in Section 4. We reconstruct the auxiliary graphs from there and change f so it is no longer tight. Essentially the same analysis shows that the subroutine can construct a new optimal solution in $O(m \cdot \mathrm{CO} + h(m+h))$ time. Since $|I(k)| = O(m)$ this leads to the weakly polynomial result that:

Theorem 3. *The scaling algorithm solves WACP in* $O((m \log C + m^2 \log r_{\max}(m + \mathrm{CO}))(m \cdot \mathrm{CO} + h(m+h)))$ *time.*

4 Solving the Restricted Problems

We now develop an algorithm to solve the restricted problems $(RASP_i)$ and $(RACP_i)$. Its outline is:

Step 1: Determine an initial RACP z^* whose support forms a chain in $\mathscr{L}^{(i)}$.
Step 2: Based on the chain $\mathrm{supp}(z^*)$, construct an auxiliary digraph G. It will turn out that $(RASP_i)$ is equivalent to a *generalized shortest path problem* (defined later) in G.
Step 3: Find a generalized s,t-path in G corresponding to an RASP p^* which uses only elements in $E^* := \{e \in E \mid z^*(e) = c_e\}$.
Step 4: If z^* and p^* are not complementary slack, i.e., if p^* is not tight on cuts in the support of z^*, update z^* along the path p^* and return to Step 2.

The algorithm maintains the invariants that $\mathrm{supp}(z^*)$ is a chain, and each cut $D \in \mathscr{L}^{(i)}$ is covered by at least one element $e \in E^*$.

4.1 Step 1: Finding an Initial RACP

Given an optimal $y^{(i)}$ for $(D^{(i)})$, we initialize $z^* = y^{(i)}$, uncrossing as necessary to obtain a chain $(\mathrm{supp}(z^*), \preceq)$ in $\mathscr{L}^{(i)}$. In a general step, let E^* denote the tight elements of z^*.

As long as $\mathscr{O}\big(\chi(E^*)\big)$ returns some $D \in \mathscr{L}^{(i)}$ with $D \cap E^* = \emptyset$, increase z^*_D until one additional constraint gets tight. Again, uncross to obtain a chain $(\mathrm{supp}(z^*), \preceq)$. Note that we need at most m oracle calls until all cuts $D \in \mathscr{L}^{(i)}$ are hit by some tight element.

4.2 Step 2: Construction of the Auxiliary Digraph

Set $\hat{\mathscr{L}}^{(i)} = \mathscr{L}^{(i)} \cup \{C_0\}$, where $C_0 = \emptyset$ is a "dummy" maximal element. We construct our auxiliary digraph $G = (\hat{V}, A, w)$ as follows: we extend chain $\mathrm{supp}(z^*)$ arbitrarily to a maximal chain $\mathscr{C} = \{C_k \prec \ldots \prec C_1 \prec C_0\}$, and assign a vertex i for each cut C_i in the chain $\mathscr{C}$. To define the arc set A, we observe that (by the modularity of $\hat{\mathscr{L}}^{(i)}$) each element $e \in E$ occurs in at least one cut $C_i \in \mathscr{C}$. Further, we observe that (by consecutivity) each element $e \in E$ occurs consecutively in $\mathscr{C}$. Thus, each element $e \in E$ uniquely defines two vertices $u_e, v_e \in \hat{V}$ such that $e \in C_{u_e} \setminus C_{u_e+1}$ and $C_{v_e+1} \setminus C_{v_e}$. We define our arc set by drawing an arc (u_e, v_e) of weight c_e for each $e \in E$. Additionally, we add a backward arc (v_e, u_e) of weight $-c_e$ for each restricted element $e \in R^{(i)}$. Let $\tilde{R}^{(i)}$ denote the added copy of $R^{(i)}$, and w the resulting weight function on the arcs. In our figures we draw the vertices $\hat{V} = \{s = k, k-1, \ldots, 1, 0 = t\}$ from bottom to top, i.e., vertex i is below vertex j iff $i > j$. Moreover, we assume that the arcs corresponding to elements in E are black, and the arcs corresponding to the restricted elements are red (see Figure 1). Thus all black arcs are directed upwards, and all red arcs are directed downwards. Also, each $C_i \in \mathscr{C}$ corresponds to the ordinary cut $\delta(\{s, k-1, k-2, \ldots, i\})$ in G. The black arcs are the outgoing arcs, and the red ("backward") arcs are the incoming arcs.

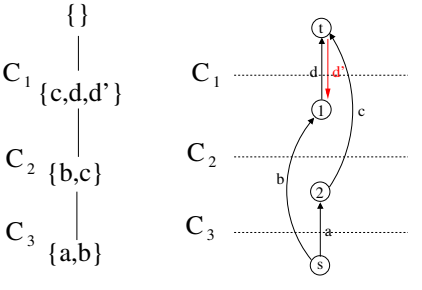


Fig. 1. Chain $(\mathscr{C}, \preceq)$ with $d \in R^{(i)}$, and auxiliary digraph $G = (\hat{V}, A)$

We also consider the shrunk graph G^* which is constructed similarly, except that it uses $\mathrm{supp}(z^*)$ instead of an extension of $\mathrm{supp}(z^*)$ as the basis of the construction. Note that G^* arises from G if we contract vertices i and $i-1$ whenever $C_i \notin \mathscr{C}$. We denote by $\tilde{G}$ and $\tilde{G}^*$ the subgraphs of G and G^* that contain only the tight arcs of G and G^*, respectively. Note that all red arcs are tight, and any s,t-path in $\tilde{G}^*$ is complementary slack with z^*. However, it is not sufficient to simply compute an s,t-path in $\tilde{G}^*$, since an s,t-path in $\tilde{G}^*$ does not necessarily correspond to a RASP. But if $\tilde{G}$ does contain an s,t-path, then each cut is hit and it is also complementary slack with z^*. However, it might be possible that $\tilde{G}$ does not contain an s,t-path despite $(RASP_i)$ having an optimal solution. Since z^* is a chain, G and G^* have $O(h)$ vertices, so constructing them costs $O(h+m)$ time.

Definition 2. *Given digraph $G = (V \cup \{t\}, A, w)$ with source $s \in V$ and sink t, a poset $P = (V, \preceq)$ (with unique minimal element $s \in V$) on V, and the collection of ideals $\mathscr{D}(P) = \{I \subseteq V \mid i \in I,\ j \preceq i \text{ implies } j \in I\}$, we define*

$$(GSP) \quad \min\{w^T p \mid p(\delta^+(I)) - p(\delta^-(I)) \geq 1,\ \forall I \in \mathscr{D}(P),\ p \geq 0\}.$$

Note that (GSP) is a special instance of a submodular flow problem so that polynomial algorithms exist. However, we provide a much simpler algorithm for this special case in Section 4.3. We call $W = \{e_1, \ldots, e_n\} \subseteq 2^A$ a *generalized s,v-path* in G, if $\text{tail}(e_1) = s$, $\text{head}(e_n) = v$ and $\text{tail}(e_{i+1}) \preceq \text{head}(e_i)$ for all $i = 2, \ldots, n$.

Lemma 4. *The incidence vector p of a generalized s,t-path W is a feasible solution of (GSP).*

Proof. Add a blue arc (u,v) of cost zero to G whenever $v \prec u$ in the poset $(V, \preceq)$ induced by $(\mathscr{L}, \preceq)$. Then any generalized s,t-path in G corresponds to an ordinary s,t-path in this extended graph using a blue arc whenever $\text{tail}(e_{i+1}) \prec \text{head}(e_i)$.

Now suppose the ideal $I \subseteq V$ is violated by p, i.e., we have $p(\delta^+(I)) - p(\delta^-(I)) < 1$. Then, since p is a $\{0,1\}$-vector, there must exist a blue arc $(u,v) \in \delta^+(I)$, meaning that $u \in I$ and $v \notin I$. However, this contradicts the property of ideals that $u \in I,\ v \prec u \Rightarrow v \in I$.

Assuming that $D \neq T$ whenever D is a lower neighbor of T in $(\mathscr{L}^{(i)}, \preceq)$ (D is a *lower neighbor* of T if $D \preceq T$ and $D \preceq S \preceq T$ implies $D = S$ or $D = T$), we use the modularity of $\mathscr{L}^{(i)}$ to show that $\mathscr{L}^{(i)}$ is a distributive lattice. Thus, by Birkhoff's Theorem [2], the cuts in the maximal chain $\mathscr{C}$ (and thus the vertices of G) correspond to a linear extension of the join-irreducible elements in the lattice $\mathscr{L}^{(i)}$ (recall that $D \in \mathscr{L}^{(i)}$ is join-irreducible if $D = S \vee T$ implies $D = S$ or $D = T$).

Theorem 4. *Let $P = (V, \preceq)$ be the induced poset on the join-irreducible cuts of $(\mathscr{L}^{(i)}, \preceq)$. Then any feasible solution $\hat{p}$ of (GSP) corresponds to a RASP p^* of the same objective value.*

Note that we can assume w.l.o.g. that $\hat{p}$ satisfies $\min\{\hat{p}(e), \hat{p}(e')\} = 0$ for each $e \in R^{(i)}$. The RASP p^* is then obtained from $\hat{p}$ by setting $p^*(e) = \hat{p}(e)$ if $e \in E \setminus R^{(i)}$, $p^*(e) = \hat{p}(e)$ if $e \in R^{(i)}$ and $\hat{p}(e) \geq 0$, and $p^*(e) = -\hat{p}(e')$ if $e' \in \tilde{R}^{(i)}$ and $\hat{p}(e') > 0$.

4.3 Step 3: Finding a Generalized Path on Tight Arcs

To find a generalized s,t-path that is complementary slack with z^*, we need to look for a generalized path in the subgraph $\tilde{G}$ consisting of the tight arcs in G. Note that $\tilde{G}$ contains a generalized s,t-path (otherwise there exists a cut $D \in \mathscr{L}^{(i)}$ not containing any tight element). Also, all generalized s,t-paths in $\tilde{G}$ have the same cost (as the arcs are tight). A breadth-first-search (BFS) tree in $\tilde{G}$ starting in s corresponds to s,v-paths to all vertices $v \in \hat{V}$ that are reachable on ordinary paths in $\tilde{G}$. We construct a generalized s,t-path in $\tilde{G}$ with the following:

Generalized path algorithm: First, apply BFS to find all vertices reachable from s on an ordinary path of tight arcs, i.e., construct a tree T in $\tilde{G}$ rooted at s. If $t \notin T$, add blue arcs from all vertices in the tree to lower neighbours in the underlying poset that are not in T. Then extend the current tree using these new arcs. Iterate until t is in the tree.

This algorithm terminates with a generalized s,t-path P. For further analysis, we extend our graphs by additional blue arcs (u,v) of cost zero whenever $v \prec u$. These blue arcs are directed downwards (as $i \prec j$ implies $i > j$ for any two vertices $i, j \in \hat{V}$.) Note that a generalized s,t-path P is an ordinary s,t-path on black, red and blue arcs in $\tilde{G}$ when extended by these blue arcs. Since any generalized path P is feasible by Lemma 4, and since it is optimal whenever it is an ordinary path in $\tilde{G}^*$ (by complementary slackness), if the generalized s,t-path P contains no blue arc in $\tilde{G}^*$, then P corresponds to a shortest RASP, and z^* corresponds to a maximum RACP. The BFS costs $O(m+h)$ time.

Recall that in order to prove that the step length θ is integer, we needed a chain $\{Q_l \prec \cdots \prec Q_0\}$ satisfying (P1) and (P2). We can show that $\mathscr{C}' = \{C \in \mathscr{C} \mid p^*(C) = 1\}$ (for the RASP p^* corresponding to the GSP P) is such a desired chain. Thus, if the generalized s,t-path P returned by the algorithm above contains no blue arc in $\tilde{G}^*$, we are done. Otherwise, the generalized path P tells us how to update the cut packing z^*.

4.4 Step 4: Update of the RACP

Let $\tilde{G}^*$ denote the shrunk subgraph of $\tilde{G}$ corresponding to the chain $\mathrm{supp}(z^*) \cup C_0 = C_r \prec \cdots \prec C_1 \prec C_0$. Let $s = v_r, \ldots, v_1, v_0 = t$ denote the vertices of $\tilde{G}^*$, $A^* \cup R^*$ denote the black and red tight arcs, B^* denote the added blue arcs, and P^* be the restriction of the generalized s,t-path P to the arcs in the shrunk graph $\tilde{G}^*$. If P^* contains a blue arc (i.e., if the RASP p^* corresponding to P and z^* does not satisfy complementary slackness), we update z^* as below.

We write $v \geq w$ if v lies below w in our representation of $\tilde{G}^*$. Note that the blue arcs of P^* correspond to the cuts in the support on which the path is not necessarily tight. Let b_1 be the first blue arc in $P^* \cap B^*$, e_1 be the first black predecessor of b_1 in $P^* \cap A^*$, and f_1 be the first black successor of b_1 in $A^* \cap P^*$ (see Figure 2). Then $u = \mathrm{tail}(f_1) \geq v = \mathrm{head}(b_1) \geq w = \mathrm{head}(e_1)$. Then for each arc $e \in A^* \cup R^* \cup B^*$, there exist exactly two vertices $j_e \leq i_e$ such that $e \in C_{j_e+1} \setminus C_{j_e}$ and $e \in C_{i_e} \setminus C_{i_e+1}$. Namely, $j_e = \mathrm{head}(e)$ and $i_e = \mathrm{tail}(e)$ if e black, and $i_e = \mathrm{head}(e)$ and $j_e = \mathrm{tail}(e)$ if e blue or red.

Sketch of the update algorithm: The problem is that C_v is in the support of z^* with $p^*(C_v) > 1$. We first update z^* by subtracting $\varepsilon = \min\{z^*_D \mid D \in \mathrm{supp}(z^*)\}$ from $z^*_{C_v}$, and adding ε to both, $z^*_{C_w}$ and $z^*_{C_{u+1}}$ without violating the arcs e_1 and f_1. However, this update might violate other arcs in $\tilde{G}^*$. We say that a black arc e is violated by z^* if $z^*(e) > c_e$, and that a red arc e' is violated if $z^*(e') < c_e$ holds. (If $z^*(e') > c_e$ for a red arc, then $z^*(e) > c_e$ also holds for its black copy e, so that we need to repair it anyway.) Note that, after this first update, a black arc e is violated exactly if for its head j_e and its tail i_e one of the following three holds: (1) $i_e < u$ and $w < j_e$, or (2) $i_e < u$ and $u \leq j_e \leq v$, or (3) $v < i_e \leq w$ and $w \leq j_e$.

Since $\tilde{G}^*$ does not contain an ordinary s,t-path (i.e., one without blue arcs), either i_e (and thus j_e) is not reachable from s on an ordinary path, or t cannot be reached from j_e on an ordinary path. In the former case, we can repair e by subtracting ε from $z^*_{C_{i_e}}$ and adding it to $z^*_{C_{i_e+1}}$. This operation could cause new arcs to be violated. However, a black arc can only be violated by this operation if its head is i_e. Moreover, since i_e is

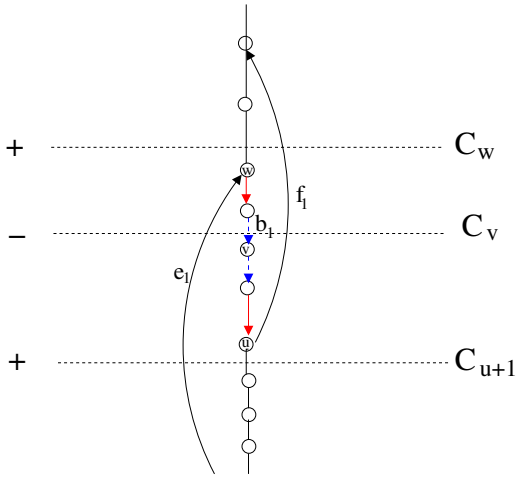


Fig. 2. First update of z^*

not reachable on an ordinary path from s, the tail of this new violated arc is also not reachable from s, and we can do the same repairing operation at its tail. This procedure terminates after at most $|\mathrm{supp}(z^*)| \leq h$ steps as i_e is not reachable from s.

In the latter case, i.e., if t cannot be reached on an ordinary path from j_e, then we can repair e at its head j_e, i.e., we subtract ε from $z^*_{C_{j_e+1}}$ and add it to $z^*_{C_{j_e}}$. Then, (similar arguments as above), these operations violate a black arc exactly if its tail is j_e, and we know that there is no ordinary path from its head to t. Thus, we can repair it with the same operation at its head. This procedure terminates after at most $|\mathrm{supp}(z^*)| \leq h$ steps as t is not reachable from j_e. In the case when a red arc is violated, we can repair it at its head i_e by adding ε to $z^*_{C_{i_e}}$ and subtracting it from $z^*_{C_{i_e+1}}$.

It is not hard to see that none of the arcs can be violated twice. Thus, after $O(m)$ operations we arrive at a feasible cut packing z^* of better objective value. Instead of choosing ε as the minimum value of *all* cuts in the support of z^* it suffices to choose ε as the minimum of all cuts on which we subtract ε. This way, we ensure that at least one cut vanishes from the support of z^*, so that we need at most $|V| = O(h)$ update calls.

Theorem 5. *This algorithm finds an optimal RASP and RACP in* $O(m \cdot \mathrm{CO} + h(m+h))$ *time.*

5 Final Thoughts

We developed the first combinatorial polynomial algorithm for Weighted Abstract Cut Packing, one important case of lattice polyhedra. There are some further questions here: Our algorithm is weakly polynomial; is there a way to make it strongly polynomial? There is another variation of lattice polyhedra in [13] with r *sub*modular and $\mathscr{L}$ being consecutive *super*modular. So far, there exists a two-phase greedy algorithm for this variant in case r is monotone [3]. Is there a combinatorial polynomial algorithm also

for submodular (but not necessarily monotone) r? Gröflin and Hoffman [9] extended the lattice polyhedra model further to $\{0, \pm 1\}$ matrices, and again showed that these were TDI. Fujishige and Peis [6] showed that when $\mathscr{L}$ is distributive this problem is equivalent to submodular flow, and so has polynomial algorithms. Can we get a combinatorial polynomial algorithm also for the general case?

References

1. Applegate, D.L., Cook, W.J., McCormick, S.T.: Integral Infeasibility and Testing Total Dual Integrality. OR Letters 10, 37–41 (1991)
2. Birkhoff, G.: Lattice Theory. Amer. Math. Soc. 91 (1991)
3. Faigle, U., Peis, B.: Two-phase greedy algorithms for some classes of combinatorial linear programs. In: Proceedings SODA 2008, pp. 161–166 (2008)
4. Ford Jr., L.R., Fulkerson, D.R.: Maximal Flow through a Network. Canadian J. of Mathematics 8, 399–404 (1956)
5. Frank, A.: Increasing the rooted connectivity of a digraph by one. Math. Programming 84, 565–576 (1999)
6. Fujishige, S., Peis, B.: Lattice Polyhedra and Submodular Flows. In: Proc. of Cologne-Twente-Workshop, CTW 2010 (2010) (to appear)
7. Fujishige, S.: Submodular Functions and Optimization, 2nd edn. North-Holland, Amsterdam (2005)
8. Fulkerson, D.R.: Blocking and Anti-Blocking Pairs of Polyhedra. Math. Prog. 1, 168–194 (1971)
9. Groflin, H., Hoffman, A.J.: Lattice Polyhedra II: Generalizations, Constructions, and Examples. Annals of Discrete Mathematics 15, 189–203 (1982)
10. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Heidelberg (1988)
11. Hoffman, A.J.: A Generalization of Max Flow-Min Cut. Math. Prog. 6, 352–359 (1974)
12. Hoffman, A.J.: On Lattice Polyhedra III: Blockers and Anti-Blockers of Lattice Clutters. Math. Prog. Study 8, 197–207 (1978)
13. Hoffman, A., Schwartz, D.E.: On lattice polyhedra. In: Hajnal, A., Sos, V.T. (eds.) Proceedings of Fifth Hungarian Combinatorial Coll, pp. 593–598. North-Holland, Amsterdam (1978)
14. Martens, M., McCormick, S.T.: A Polynomial Algorithm for Weighted Abstract Flow. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO 2008. LNCS, vol. 5035, pp. 97–111. Springer, Heidelberg (2008)
15. McCormick, S.T.: A Polynomial Algorithm for Abstract Maximum Flow. Extended abstract. In: Proceedings of the 7th ACM-SIAM SODA, pp. 490–497 (1995)
16. McCormick, S.T.: Submodular Function Minimization. In: Aardal, K., Nemhauser, G., Weismantel, R. (eds.) Handbook on Discrete Optimization, ch. 7, pp. 321–391. Elsevier, Amsterdam (2006)
17. McCormick, S.T., Fujishige, S.: Strongly Polynomial and Fully Combinatorial Algorithms for Bisubmodular Function Minimization. Mathematical Programming 122, 87–120 (2010)
18. Schrijver, A.: Combinatorial Optimization. Springer, Heidelberg (2003)
19. Schrijver, A.: Theory of Linear and Integer Programming. John Wiley & Sons, New York (1986)
20. Seymour, P.D.: Decomposition of Regular Matroids. JCT B 28, 305–359 (1980)

Convexification Techniques for Linear Complementarity Constraints*

Trang T. Nguyen[1], Mohit Tawarmalani[2], and Jean-Philippe P. Richard[1]

[1] Department of Industrial and Systems Engineering, University of Florida
[2] Krannert School of Management, Purdue University

Abstract. We develop convexification techniques for linear programs with linear complementarity constraints (LPCC). In particular, we generalize the reformulation-linearization technique of [9] to complementarity problems and discuss how it reduces to the standard technique for binary mixed-integer programs. Then, we consider a class of complementarity problems that appear in KKT systems and show that its convex hull is that of a binary mixed-integer program. For this class of problems, we study further the case where a single complementarity constraint is imposed and show that all nontrivial facet-defining inequalities can be obtained through a simple cancel-and-relax procedure. We use this result to identify special cases where McCormick inequalities suffice to describe the convex hull and other cases where these inequalities are not sufficient.

1 Introduction

In this paper, we consider complementarity problems of the form:

$$S^{\perp} = \left\{(x, y, z) \in \mathbb{R}^m \times \mathbb{R}^n_+ \times \mathbb{R}^n_+ \mid (x, y, z) \in S, y \perp z\right\},$$

where $y \perp z$ signifies that $y_i z_i = 0$ for all $i \in N = \{1, \ldots, n\}$.

Complementarity problems have numerous engineering applications including traffic equilibrium, structural mechanics, and structural design. Complementarity problems have been used to model optimal control problems for multiple robot systems. In economics, complementarities arise when expressing conditions for Nash equilibria, invariant capital stock, spatial price equilibrium, and game theoretic models. We refer to [4] for a survey of applications of $S^{\perp}$.

A particularly important subcase of the complementarity problem arises when the set S is chosen to be a polyhedron. Such a choice gives rise to problems often referred to as *linear programs with complementarity constraints* (LPCC) which are concerned with optimizing a linear function over the following set:

$$S^{\perp}_L = \left\{ (x, y, z) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \,\middle|\, \begin{array}{l} Ax + By + Cz \geq d \\ y \perp z \end{array} \right\},$$

where $A \in \mathbb{R}^{p \times m}$, $B \in \mathbb{R}^{p \times n}$, $C \in \mathbb{R}^{p \times n}$, $d \in \mathbb{R}^p$ and where we assume that the constraints $y_i \geq 0$ and $z_i \geq 0$ are implicitly satisfied by feasible solutions

* This work was supported by NSF CMMI grants 0856605 and 0900065.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 336–348, 2011.

to $Ax + By + Cz \geq d$. In [5], it is shown that LPCCs have applications in multilevel optimization, inverse convex quadratic programming, cross-validated support vector regression, indefinite quadratic programming, piecewise linear programming, and quantile minimization.

Complementarity constraints often occur through the use of KKT conditions. For instance, consider the case of a Stackelberg game in which a single leader optimizes her set of decisions in a system where several followers will optimize their plans depending on her decisions. Such a problem has a natural multilevel (or hierarchical) formulation. If the optimization problems solved by the followers are linearly constrained quadratic convex problems, a simple way of formulating the problem into a traditional mathematical program is to replace the followers' optimization models by their corresponding KKT systems. Complementary slackness conditions in the KKT systems then reduce to (linear) complementarity constraints between primal and dual variables while gradient conditions provide additional (linear) relations between primal and dual variables.

Polyhedral studies of problems with complementarity constraints can be found in [3] for the complementarity knapsack problem and in [13] for relaxations of KKT systems arising in quadratic programming. The reformulation-linearization technique has been used to convexify linear complementarity problems, which form a subclass of LPCCs [10]. Recently [5] derived new relaxations for LPCCs and compared them numerically.

In this paper, we study how integer programming convexification techniques can be extended to derive convex hulls of LPCCs. In Section 2, we present an extension of the reformulation-linearization technique (RLT) of [9] that can be used to sequentially convexify LPCCs. The extension is different from traditional RLT in that it does not use polynomial factors. We also show it to be stronger than the direct applications of RLT proposed in [5]. In Section 3, we study a particular class of LPCCs that appears naturally in optimality conditions of mathematical programs with linear constraints due to complementary slackness and primal feasibility requirements. In this case, our enhanced RLT scheme reduces to the traditional RLT. However, for the case where a single cardinality constraint is considered with k side constraints, we show that nontrivial RLT inequalities can be interpreted as obtained through a cancel-and-relax approach in which up to $k-1$ linear constraints are aggregated with the complementarity constraint and McCormick relaxations are used to linearize the resulting bilinear inequality. In Section 4, we use this cancel-and-relax procedure to obtain closed-formed description of the convex hull of sets with a single complementarity constraint and few side constraints. We give concluding remarks in Section 5.

2 Reformulation-Linearization Technique

In this section, we develop convexification techniques for $S_L^{\perp}$. We denote $\{x_i\}_{i\in I}$ by x_I. Given a set S, we define its convex hull by $\text{conv}(S)$ and its projection onto the space of x-variables by $\text{proj}_x(S)$. In [6], Jeroslow developed a seminal convexification procedure for $S_L^{\perp}$ that we summarize next.

Theorem 1 ([6]). $\text{conv}(S_L^{\perp}) = \text{conv}\big(\bigcup_{I \subseteq N} \big(S_L^{\perp} \cap \{(y,z) \mid y_I = 0, z_{N\setminus I} = 0\}\big)\big)$.

Here, $\text{conv}(S_L^{\perp})$ is expressed as the convex hull of a union of polyhedral sets. If these polyhedral sets share the same recession cone (in particular, if they are bounded), it follows that $\text{conv}(S_L^{\perp})$ is a polyhedral set. It is well-known that 0-1 mixed-integer programs are a special case of LPCC. In this paper, we extend the convexification techniques developed for 0-1 mixed-integer programs (BMIPs) to LPCCs. For ease of presentation, we assume that $S_L^{\perp}$ is compact and the polyhedral sets in Theorem 1 whose disjunctive union is $S_L^{\perp}$ are non-empty.

First, we extend the reformulation linearization technique (RLT) initially developed for BMIPs to LPCCs. In [10], RLT is used to solve the linear complementarity problem, a special case dealing with the feasibility problem over linear complementarity constraints. To allow the use of RLT, the authors reformulate the problem as a BMIP. This approach can also be applied to LPCCs by first transforming the problem into a BMIP (see [5] for example) and then employing the standard RLT scheme. However, such a scheme is not completely satisfactory in that it requires explicit bounds on the y and z variables and adds n binary variables to the formulation. The scheme we present next circumvents these difficulties and has a direct interpretation in the original space of variables.

We describe the extended technique below. First, we define $q_i = \frac{y_i}{y_i+z_i}$ and $\bar{q}_i = 1 - \frac{y_i}{y_i+z_i} = \frac{z_i}{y_i+z_i}$. We also let $q(I,J) = \prod_{i\in I} q_i \prod_{j\in J} \bar{q}_j$. The technique at the k^{th} level is composed of a reformulation and a linearization step. For the reformulation step, we multiply $Ax+By+Cz \geq d$ with each $q(I,J)$ such that $|I|+|J| = k$. Since $q_i\bar{q}_i = 0$ we may assume that $I \cap J = \emptyset$. The reformulation contains nonlinear terms $q(I,\emptyset)$, $x_i q(I,\emptyset)$, $y_j q(I,\emptyset)$ and $z_j q(I,\emptyset)$. In the linearization step, we replace these nonlinear expressions with the new variables q_I, $u(i,I)$, $v(j,I)$ and $w(j,I)$ respectively. For $a \in \{x_i, y_j, z_j\}$ note that $aq(I,J)$ can be linearized as $\sum_{J'\subseteq J}(-1)^{|J'|}\prod_{i\in I\cup J'} aq_i = \sum_{J'\subseteq J}(-1)^{|J'|} aq(I\cup J', \emptyset)$. Additionally, observe that $v(j,I) = v(j, I\setminus\{j\})$ and $z_j q(I,J) = z_j q(I, J\setminus\{j\})$. Further, it follows from $y_j z_j = 0$ that $y_j q(I,J) = 0$ if $j \in J$ and $z_j q(I,J) = 0$ if $j \in I$. We may also relax the relation $v(i,I) = v(i', I\cup\{i\}\setminus\{i'\})\frac{y_i+z_i}{y_{i'}+z_{i'}}$ where $i \notin I$.

The above product factors can be derived from inclusion certificates for the positive orthant [12]. Inclusion certificates allow the generalization of RLT to convexification of disjunctive union of convex sets [11]. However, the departure here is that the product factors have a simple algebraic form, which is typically not the case in [11]. Denoting by $\text{RLT}^k(S)$, the k^{th} level extended RLT relaxation of S, we obtain the following sequential convexification result.

Theorem 2. *Let* $S = \{(x,y,z) \mid Ax + By + Cz \geq d\}$ *and* $S_j = S \cap \{(x,y,z) \mid y_i z_i = 0 \, \forall i \leq j\}$. *Then,* $\text{proj}_{x,y,z}\big(\text{RLT}^1(\text{conv}(S_j) \cap \{(x,y,z) \mid y_{j+1}z_{j+1} = 0\})\big) = \text{conv}(S_{j+1})$. *Therefore, n iterations of the procedure yields* $\text{conv}(S_n)$. *In particular,* $\text{proj}_{x,y,z}\big(\text{RLT}^n(S \cap \{(x,y,z) \mid y_i z_i = 0 \text{ for } i = 1,\ldots,n\})\big) = \text{conv}(S_n)$.

Our extension of RLT can also be viewed as an application of the standard RLT scheme with the additional constraints $y_i = q_i y_i$ and $z_i = (1-q_i)z_i$ where q_i is binary. However, our approach allows the interpretation of q_i as $\frac{y_i}{y_i+z_i}$.

This interpretation has advantages. For example, it allows us to relate $v(i, I)$ to $v(i', I \cup \{i\})$ as mentioned before. We will also show in Example 1 that the standard RLT relaxation obtained by using y_i and z_i as product factors (as suggested in [5]) does not yield the desirable properties of convexification given in Theorem 2. In the above, $\text{proj}_{x,y,z}(\text{RLT}^1(\text{conv}(S_j) \cap \{(x, y, z) \mid y_{j+1}z_{j+1} = 0\}))$ can be viewed as an extension of the lift-and-project operator to LPCC [2]. It is clear that one can also extend the Lovász and Schrijver operator by requiring that $[1\, q\, x\, y\, z]^\intercal [1\, q\, x\, y\, z] \succeq 0$, where we assume q, x, y, and z are row vectors.

Example 1. Consider the LPCC:

$$\begin{aligned}
\min\ & 7x_1 - 5x_2 - x_3 + 8y_1 - 4z_1 \\
\text{s.t. } & -9x_1 - 2x_2 + 3x_3 - 6y_1 + 2z_1 \le -4 && (2.1)\\
& 2x_1 + 8x_3 - 2y_1 + 3z_1 \le 5 && (2.2)\\
& 9x_1 + 2x_2 - 4x_3 - 4y_1 + 6z_1 \le 4 && (2.3)\\
& 7x_1 + 6x_2 + 2x_3 - 6y_1 + z_1 \le 5 && (2.4)\\
& 9x_1 + 6x_2 - x_3 - 4z_1 \le 4 && (2.5)\\
& 0 \le x_1, x_2, x_3, y_1, z_1 \le 1 && (2.6)\\
& y_1 z_1 = 0.
\end{aligned}$$

As expected from Theorem 2, the following relaxation obtained by the above RLT scheme yields the optimal objective function value of 0.2222:

$$\begin{aligned}
\min\ & 7x_1 - 5x_2 - x_3 + 8y_1 - 4z_1 \\
\text{s.t. } & -9u_1 - 2u_2 + 3u_3 - 6y_1 \le -4q \\
& -9(x_1 - u_1) - 2(x_2 - u_2) + 3(x_3 - u_3) + 2z_1 \le -4(1 - q) \\
& \max\{x_1 + q - 1, 0\} \le u_1 \le \min\{x_1, q\} \\
& 0 \le y_1 \le q, \quad 0 \le z_1 \le 1 - q \\
& F_1 x + F_2 u + F_3 y_1 + F_4 z_1 + F_5 q \le f,
\end{aligned}$$

where we only used (2.1) and $0 \le x_1 \le 1$ to illustrate these RLT operations and we represented by $F_1 x + F_2 u + F_3 y_1 + F_4 z_1 + F_5 q \le f$ all linearized constraints that can be obtained similarly from multiplying (2.2)-(2.5) and $0 \le x_2, x_3 \le 1$ with q and $1 - q$. In [5], the authors recommend the use of the product factors y_1 and z_1 for the above problem while exploiting $y_i z_i = 0$. We show that these product factors are insufficient for convexification. In addition, we retain the linear constraints from the original formulation to make the relaxation tighter. In particular, this results in the linearization of the following formulation:

$$\begin{aligned}
\min\ & 7x_1 - 5x_2 - x_3 + 8y_1 - 4z_1 \\
\text{s.t. } & (2.1), (2.2), (2.3), (2.4), (2.5), (2.6) \\
& -9x_1 y_1 - 2x_2 y_1 + 3x_3 y_1 - 6y_1^2 \le -4y_1 \\
& -9x_1 z_1 - 2x_2 z_1 + 3x_3 z_1 + 2z_1^2 \le -4z_1 \\
& 0 \le x_1 y_1 \le y_1, \quad 0 \le x_1 z_1 \le z_1 \\
& 0 \le y_1^2 \le y_1, \quad 0 \le z_1^2 \le z_1 \\
& F_1' y_1 + F_2' z_1 + F_3' y_1 x + F_4' z_1 x + F_5' y_1^2 + F_6' z_1^2 \le 0,
\end{aligned}$$

where $F_1' y_1 + F_2' z_1 + F_3' y_1 x + F_4' z_1 x + F_5' y_1^2 + F_6' z_1^2 \le 0$ represent all constraints that can be obtained similarly from multiplying (2.2)-(2.5) and $0 \le x_2, x_3 \le 1$

with y_1 and z_1. Here, the linearization replaces the bilinear and square terms with new variables to produce an LP relaxation. The optimal objective value for this relaxation is -3.3137 showing the weakness of the approach that does not use the correct product factors. Further, introducing the McCormick constraints for all bilinear products only improves the optimal objective function value to -0.9630. Even if one multiplies Constraints (2.1)-(2.5) with $(1 - y_1)$ and $(1 - z_1)$ and linearizes them, the lower bound does not improve.

We conclude this section by pointing out some connections between our extended RLT scheme and traditional RLT for BMIPs. BMIPs are a special case of LPCCs [6]. Consider $M = \{Ax + By \leq d, y \in \{0,1\}^n\}$. Then, M can be reformulated as: $M' = \{Ax + By \leq d, y \geq 0, z \geq 0, z_i + y_i = 1, z_i y_i = 0\}$. Now, consider the RLT scheme described in Section 2 applied to M'. Since $y_i + z_i = 1$, $q(I, J) = \prod_{i \in I} y_i \prod_{j \in J}(1 - y_j)$. Further, $v(j, I) = v(j, I \backslash \{j\})$ is the familiar relation $y_j^2 = y_j$. The relation $y_i y_j = y_j y_i$ follows from relaxing $v(i, I) = v(i', I \cup \{i\} \backslash \{i'\}) \frac{y_i + z_i}{y_{i'} + z_{i'}}$. Finally, the relation $v(i, I) = q(I \cup \{i\})$ follows from the linearized relation obtained upon multiplying $z_i + y_i = 1$ with $q(I \cup \{i\}, \emptyset)$. This shows that our extended RLT scheme is a natural generalization of that initially described for BMIPs.

3 A Class of Complementarity Problems

We study the following special case of $S_L^{\perp}$ that occurs when considering primal constraints and (a subset of) complementary slackness conditions from a KKT system associated with a constrained nonlinear optimization problem:

$$\bar{S} = \left\{ (x, y) \in [0,1]^m \times [0,1]^n \middle| \begin{array}{l} y_j g_j(x) = 0, \forall j = 1, \ldots, n \\ g_j(x) \geq 0, \quad \forall j = 1, \ldots, k \end{array} \right\}$$

where $g(x) = (g_1(x), \ldots, g_k(x)) : \mathbb{R}^m \to \mathbb{R}^k$ and $k \geq n$. We assume that $g(x)$ are continuous functions. Therefore, $\bar{S}$ is compact and so is its convex hull [8]. We refer to $y_j g_j(x)$ as *complementarity constraints*. By rescaling variables, we assume without loss of generality that the variables belong to the unit hypercube. Note that not all constraints $g_j(x) \geq 0$ in $\bar{S}$ have an associated complementarity constraint. Observe also that, in $\bar{S}$, the variables y_j through which complementarity is imposed do not appear otherwise in the constraint functions g. Therefore, we refer to $\bar{S}$ as a separable complementarity problem (SCP).

Proposition 1. *If (x, y) is an extreme point of $\bar{S}$ then $y \in \{0,1\}^n$.*

Since the convex hull of a compact set is the convex hull of its extreme points [7], it follows that $\text{conv}(\bar{S}) = \text{conv}(\bar{S} \cap \{(x, y) \mid y \in \{0,1\}^n\}$. Therefore, if $g(x)$ is a vector of linear functions, SCP can be convexified using the standard RLT approach with product-factors y_i and $1 - y_i$. Next, we study RLT further for special cases of $\bar{S}$ where functions $g_j(x)$ are linear for $j = 1, \ldots, k$.

Let $M = \{1, \ldots, m\}$. The sets we consider are of the form:

$$S^{n,k} = \left\{ (x,y) \in [0,1]^{m+n} \;\middle|\; \begin{array}{ll} y_j(\sum_{i \in M} \alpha_i^j x_i - \beta_j) = 0, & \forall j = 1, \ldots, n \\ \sum_{i \in M} \alpha_i^j x_i - \beta_j \geq 0, & \forall j = 1, \ldots, k \end{array} \right\}.$$

To streamline the discussion, we will only consider cases where the right-hand-sides of the linear functions are 0. We show next in Proposition 2 that this assumption is without loss of generality as the facet-defining inequalities of $S^{n,k}$ can be obtained from the following homogenized variant:

$$S_0^{n,k} = \left\{ (x_0,x,y) \in [0,1]^{1+m+n} \;\middle|\; \begin{array}{ll} y_j(\sum_{i \in M} \alpha_i^j x_i - \beta_j x_0) = 0, & \forall j = 1, \ldots, n \\ \sum_{i \in M} \alpha_i^j x_i - \beta_j x_0 \geq 0, & \forall j = 1, \ldots, k \end{array} \right\},$$

where $\alpha_i^j \in \mathbb{R}$ for all $j = 1, \ldots, k$ and $i = 1, \ldots, m$ and $\beta_j \in \mathbb{R}$ for $j = 1, \ldots, k$. The following result establishes that $\text{conv}(S^{n,k})$ is a face of $\text{conv}(S_0^{n,k})$.

Proposition 2. *For the sets $S^{n,k}$ and $S_0^{n,k}$ defined above,*

$$\text{conv}(S^{n,k}) = \text{conv}(S_0^{n,k}) \cap \{(x_0,x,y) \in [0,1]^{1+m+n} \mid x_0 = 1\}.$$

Proposition 2 allows us to restrict our attention to:

$$C^{n,k} = \left\{ (x,y) \in [0,1]^{m+n} \;\middle|\; \begin{array}{ll} y_j(\sum_{i \in M} \alpha_i^j x_i) = 0, & \forall j = 1, \ldots, n \\ \sum_{i \in M} \alpha_i^j x_i \geq 0, & \forall j = 1, \ldots, k \end{array} \right\},$$

where $\alpha_i^j \in \mathbb{R}$ for all $j = 1, \ldots, k$ and $i = 1, \ldots, m$.

Proposition 3. *$C^{n,k}$ is full-dimensional if and only if $C^{0,k}$ is full-dimensional.*

Proposition 4. *If $\alpha_1^j > 0$ for $j \in \{1, \ldots, k\}$ then $C^{n,k}$ is full-dimensional.*

Henceforth, we assume

Assumption 1. *There exists $i \in M$ such that $\alpha_i^j > 0$ for $j \in \{1, \ldots, k\}$.*

This assumption is without loss of generality. In fact, consider the case where for each $i \in M$, there is a $k_i \in \{1, \ldots, k\}$ such that $\alpha_i^j \leq 0$. Then, construct

$$C'^{n,k} = \left\{ (x_0,x,y) \in [0,1]^{1+m+n} \;\middle|\; \begin{array}{ll} y_j(x_0 + \sum_{i \in M} \alpha_i^j x_i) = 0, & \forall j = 1, \ldots, n \\ x_0 + \sum_{i \in M} \alpha_i^j x_i \geq 0, & \forall j = 1, \ldots, k \end{array} \right\},$$

and observe that $C'^{n,k}$ satisfies Assumption 1. Further, $\text{conv}(C^{n,k})$ can be easily obtained from $\text{conv}(C'^{n,k})$ as shown next.

Proposition 5. $\text{conv}(C^{n,k}) = \text{conv}(C'^{n,k}) \cap \{(0, x, y) \in \mathbb{R}^{1+m+n}\}$.

The following result shows that facets obtained by ignoring some of the complementarity constraints often yield facets for problems where complementarity constraints are enforced.

Proposition 6. *Consider* $n' < n$ *and let* $J = \{n'+1, \ldots, n\}$. *Define* $D(J) = \left\{(x, y) \;\middle|\; y_j(\sum_{i \in M} \alpha_i^j x_i) = 0 \, \forall j \in J, (y_j)_{j \in J} \in \{0,1\}^{|J|}\right\}$. *Then,*

$$\text{conv}(C^{n,k}) = \text{conv}\left(\left(\text{conv}(C^{n',k}) \times [0,1]^{|J|}\right) \cap D(J)\right).$$

Any facet-defining inequality of $\text{conv}(C^{n',k}) \times [0,1]^{|J|}$ *which is tight at a point* $(x', y') \in \text{conv}(C^{n',k}) \times [0,1]^{|J|}$ *such that* $\sum_{i \in M} \alpha_i^j x_i' = 0$ *for* $j \in J$ *defines a facet of* $\text{conv}(C^{n,k})$. *Conversely, any facet-defining inequality* $ax + by \leq c$ *of* $\text{conv}(C^{n,k})$ *is facet-defining for* $\text{conv}(C^{n',k})$ *if* $b_j = 0$ *for* $j \in J$.

The result of Proposition 6 can be interpreted as a *zero-lifting* result where strong inequalities for a problem restriction can be trivially extended into strong inequalities for a more complex set. Nontrivial lifting results can also be developed to extend inequalities for sets containing few complementarity constraints to sets containing multiple complementarities. The following example gives an illustration of this observation by showing that facets of $C^{2,2}$ can be obtained by lifting facet-defining inequalities for $C^{1,2}$.

Example 2. Consider the set

$$C^{2,2} = \left\{ (x, y) \in [0,1]^5 \;\middle|\; \begin{array}{l} y_1(x_1 + x_2 - 2x_3) = 0 \\ y_2(2x_1 - x_2 - x_3) = 0 \\ x_1 + x_2 - 2x_3 \geq 0 \\ 2x_1 - x_2 - x_3 \geq 0 \end{array} \right\}.$$

Clearly, $-2(y_1x_1 + y_1x_2 - 2y_1x_3) + y_1(2x_1 - x_2 - x_3) \geq 0$ is valid for $C^{2,2}$. Using McCormick over- and under-estimators, this inequality can be linearized into $x_2 - x_3 + y_1 \leq 1$. The inequality is facet-defining for $C^{2,2} \cap \{y_2 = 1\}$ (which is a face of $C^{1,2} \times \{1\}$) as can be verified through the tight points (expressed as (x_1, x_2, x_3, y_1)) $(0,0,0,1)$, $(\frac{1}{2}, 1, 0, 0)$, and $(1,1,1,1)$. Considering $x_2 - x_3 + y_1 \leq 1$ as a seed-inequality one can obtain other facets for $C^{2,2}$ through lifting. In particular, the seed inequality can be lifted into $x_2 - x_3 + y_1 + \delta(2x_1 - x_2 - x_3) + \theta(1 - y_2) \leq 1$. It can be verified that choosing $\delta = 1$, $\theta = -1$ results in the lifted inequality $2x_1 - 2x_3 + y_1 + y_2 \leq 2$, which is valid for $C^{2,2}$. This inequality is facet-defining for $C^{2,2}$ as it is tight at $(x_1, x_2, x_3, y_1, y_2) = (1, 0, \frac{1}{2}, 1, 0)$ and $(1,0,0,0,0)$ in addition to the previously described points. In fact, for this set, it can be verified that the only inequality that is not also a facet of $C^{2,2} \cap \{y_2 = 0\}$ or $C^{2,2} \cap \{y_1 = 0\}$ is the inequality we just derived using lifting.

3.1 Convex Hull of One-Complementarity Sets

Proposition 6 and Example 2 indicate that sets with a single complementarity constraint, which we call *one-complementarity sets*, play a fundamental role in understanding the structure of more complex multi-complementarity sets. Subsequently, we will argue that a recursive application of the techniques developed in this section yields $\text{conv}(C^{n,k})$. For this reason, we next investigate the polyhedral structure of one-complementarity sets.

We consider sets with a single complementarity constraint and linear side constraints. In this case, RLT or lift-and-project yield the same relaxation which can also be viewed as a direct application of disjunctive programming. Therefore, in this section, we will use a disjunctive programming viewpoint for studying convexification. We consider the one-complementarity set with k linear constraints defined as follows:

$$C^{1,k} = \left\{ (x,y) \in [0,1]^{m+1} \left| \begin{array}{l} y(\sum_{i\in M} \alpha_i^1 x_i) = 0, \\ \sum_{i\in M} \alpha_i^j x_i \geq 0, \ \forall j \in K \end{array} \right. \right\},$$

where $K = \{1, \ldots, k\}$. From Proposition 1 and the lift-and-project procedure [2], the convex hull of $C^{1,k}$ is the set of (x,y) for which there exists $\bar{v}$ such that

$$\sum_{i\in M} \alpha_i^j \bar{v}_i \geq 0, \ \forall j \in K \qquad (a_j)$$

$$\sum_{i\in M} \alpha_i^1 (x_i - \bar{v}_i) = 0 \qquad (b_1)$$

$$\sum_{i\in M} \alpha_i^j (x_i - \bar{v}_i) \geq 0, \ \forall j \in K \setminus \{1\} \qquad (b_j)$$

$$\max(0, x_i - y) \leq \bar{v}_i \leq \min(1 - y, x_i), \ \forall i \in M, \qquad (r,s,t,u)$$

where the variables in parenthesis denote the corresponding dual variables. We use w to denote (a,b,r,s,t,u). In order to produce a description of $\text{conv}(C^{1,k})$ in the space of original variables (x,y), we project out the variables $\bar{v}$. This can be done using the extreme rays of the projection cone:

$$W = \left\{ w \left| \sum_{j\in K} (a_j - b_j)\alpha_i^j + r_i + s_i = t_i + u_i, \forall i \in M, \ a, \{b_j\}_{j=2}^k, r, t, u, s \geq 0 \right. \right\}.$$

It is easily verified that the inequality

$$\sum_{i\in M} \left(\sum_{j\in K} b_j \alpha_i^j + u_i - s_i \right) x_i + \sum_{i\in M} (s_i - t_i) y \geq - \sum_{i\in M} t_i$$

is facet-defining for $\text{conv}(C^{1,k})$ only if w is an extreme ray of W, see [1]. We now identify some properties of the extreme rays of the projection cone W. We say that a facet-defining inequality of $\text{conv}(C^{1,k})$ is non-trivial if it is not a multiple of $\sum_{i\in M} \alpha_i^j x_i \geq 0$ or the bounds constraints $(x,y) \in [0,1]^{m+1}$.

Proposition 7. *If w is an extreme ray of W and corresponds to a non-trivial facet of* $\text{conv}(C^{1,k})$*, then (i) at most one of the variables (r_i, s_i, t_i, u_i) is positive for all $i \in M$, (ii) $a_j b_j = 0$ for all $j \in K$, and (iii) $b_1 < 0$.*

Proposition 7 implies that (r, s, t, u) can be treated as slack variables and we need to characterize the extreme points of:

$$\overline{W} = \left\{ w' \in \mathbb{R}^{2k}_+ \;\middle|\; \begin{array}{l} \sum_{j \in K} (a_j - b_j)\alpha_i^j \leq 0,\ i \in M_1 \\ \sum_{j \in K} (a_j - b_j)\alpha_i^j \geq 0,\ i \in M \backslash M_1 \\ b_1 = -1 \end{array} \right\},$$

where w' denotes $(a, -b_1, b_2, \ldots, b_k)$.

Theorem 3. *Define*

$$\widehat{W} = \left\{ w' \in \mathbb{R}^{2k}_+ \;\middle|\; b_1 = -1, \exists T \subseteq M, |T| \leq k-1, \sum_{j \in K} (a_j - b_j)\, \alpha_i^j = 0, \forall i \in T \right\}.$$

Then, any extreme ray of W that corresponds to a non-trivial facet of $\text{conv}(C^{1,k})$ *belongs to $\widehat{W}$.*

The interpretation of Theorem 3 is that non-trivial inequalities for $\text{conv}(C^{1,k})$ are derived using a simple *cancel-and-relax* procedure. In this procedure, the j^{th} inequality is multiplied with either $a_j y$ or $b_j(1-y)$ and the resulting inequalities are added together. We can arrange for the coefficients of up to $k-1$ bilinear variables to sum to zero, i.e. to *cancel*. The remaining bilinear terms can then be over-/under-estimated, i.e. *relaxed*, using McCormick's envelopes.

In fact a recursive application of the cancel-and-relax procedure yields the convex hull of $C^{n,k}$. The following example illustrates the basic ideas.

Example 3. Consider the set of Example 2 for which we derived the inequality $2x_1 - 2x_3 + y_1 + y_2 \leq 2$. We next provide an alternate derivation of this inequality. Note that $y_2(x_2 - x_3 + y_1) - y_2 x_0 + (1 - y_2)(x_1 - 2x_3 + y_1) - (1 - y_2)x_0 + y_2(2x_1 - x_2 - x_3) \leq 0$ implies $x_1 + x_1 y_2 - 2x_3 + y_1 \leq x_0$ from which we obtain $2x_1 - 2x_3 + y_1 + y_2 \leq 1 + x_0$. When intersected with $x_0 = 1$ we obtain the desired facet-defining inequality. Observe that $x_2 - x_3 + y_1 \leq 1$ and $x_1 - 2x_3 + y_1 \leq 1$ are facet-defining inequalities for $C^{1,2}$ that can be obtained with a first application of the cancel-and-relax procedure and $y_2(2x_1 - x_2 - x_3) = 0$ is used with a negative multiplier. The variable x_0 is introduced to homogenize the inequalities of $C^{1,2}$ as assumed in the description of the cancel-and-relax procedure.

4 Explicit Inequalities via Convexification of Simple Sets

In this section, we illustrate the cancel-and-relax procedure by providing closed-form expressions for the convex hulls of $C^{1,1}$ and $C^{1,2}$. In Section 4.1, we find that $\text{conv}(C^{1,1})$ is obtained by traditional factorable relaxations. In Section 4.2, we characterize $\text{conv}(C^{1,2})$ using the cancel-and-relax principle of Section 3 and show that it yields improvements over the factorable relaxation techniques.

4.1 Convex Hull of $C^{1,1}$

We consider $C^{1,1} = \left\{(x,y) \in [0,1]^{m+1} \mid y(\sum_{i\in M} \alpha_i x_i) = 0,\ \sum_{i\in M} \alpha_i x_i \geq 0\right\}$, where $\alpha_i \in \mathbb{R}$ for $i \in M$. We assume that there exists $i \in M$ such that $\alpha_i \neq 0$ (otherwise $C^{1,1} = [0,1]^{m+1}$.) We define the sets $I^+ = \{i \in M \mid \alpha_i > 0\}$ and $I^- = \{i \in M \mid \alpha_i < 0\}$. The observation following Proposition 1 implies that the convex hull of $C^{1,1}$ is polyhedral. We also observe that, when $\text{conv}(C^{1,1})$ is full-dimensional, Assumption 1 is naturally satisfied.

Proposition 8. $\text{conv}(C^{1,1})$ *is full-dimensional if and only if* $I^+ \neq \emptyset$.

Let $I^0 = \{i \in M \mid \alpha_i = 0\}$ and define $K^{1,1} = \{(x,y) \in \mathbb{R}^{m-|I_0|+1} \mid (x,0,y) \in C^{1,1}\}$. Clearly $C^{1,1} = K^{1,1} \times [0,1]^{|I_0|}$. It follows that $\text{conv}(C^{1,1}) = \text{conv}(K^{1,1}) \times [0,1]^{|I_0|}$. Therefore, we restrict attention to sets of the form $K^{1,1}$ by assuming that $I_0 = \emptyset$, *i.e.*, $M = I^+ \cup I^-$. The inequalities that describe $\text{conv}(C^{1,1})$ can be obtained using Theorem 3. Since in this case Theorem 3 requires cancellation of zero bilinear terms, we conclude that all nontrivial inequalities of $\text{conv}(C^{1,1})$ can be obtained through McCormick relaxation of the bilinear inequality with $b_1 = -1$. Since $\max\{0, x_i + y - 1\} \leq yx_i \leq \min\{x_i, y\}$, it follows that

$$\sum_{i\in I^+} \alpha_i \max\{0, x_i + y - 1\} + \sum_{i\in I^-} \alpha_i \min\{x_i, y\} \leq 0$$

is a valid inequality for $\text{conv}(C^{1,1})$. Although this inequality is nonlinear, it admits a reformulation with $2^{|M|}$ linear inequalities, or a polynomial-sized linear reformulation in a lifted space. The former reformulation is obtained by considering all subsets S^+ of I^+ and all subsets S^- of I^-, by replacing $\max\{0, x_i + y - 1\}$ with 0 for $i \in I^+ \backslash S^+$ and with $x_i + y - 1$ for $i \in S^+$ and by substituting $\min\{x_i, y\}$ with x_i for $i \in S^-$ and with y for $i \in I^- \backslash S^-$. It leads to the following exponential set of linear inequalities

$$\left(\sum_{i\in S^+} \alpha_i + \sum_{i\in I^-\backslash S^-} \alpha_i\right) y + \sum_{i\in S^+} \alpha_i x_i + \sum_{i\in S^-} \alpha_i x_i \leq \sum_{i\in S^+} \alpha_i \tag{4.7}$$

where $S^+ \subseteq I^+$ and $S^- \subseteq I^-$. It is clear that inequalities (4.7) are valid for $\text{conv}(C^{1,1})$. We obtain the following result as a consequence of Theorem 3.

Theorem 4. $\text{conv}(C^{1,1}) = Q$ *where*

$$Q = \left\{ (x,y) \;\middle|\; \begin{array}{l} 0 \leq x_i \leq 1, \forall i \in N \\ 0 \leq y \leq 1, \\ \sum_{i\in M} \alpha_i x_i \geq 0 \\ \left(\sum_{i\in S^+} \alpha_i + \sum_{i\in I^-\backslash S^-} \alpha_i\right) y + \sum_{i\in S^+} \alpha_i x_i + \sum_{i\in S^-} \alpha_i x_i \leq \sum_{i\in S^+} \alpha_i \\ \qquad\qquad \forall (S^+, S^-) \subseteq (I^+, I^-) \end{array} \right\}.$$

Although the linear description of $\text{conv}(C^{1,1})$ we give in Theorem 4 is exponential in size, its inequalities can be separated in polynomial time. Not all inequalities presented in Theorem 4 are strong for $\text{conv}(C^{1,1})$. We next derive necessary and sufficient conditions for these inequalities to be facet-defining.

Proposition 9. *The following inequalities are facet-defining for* $\text{conv}(C^{1,1})$*: (i)* $y \geq 0$*, (ii)* $\sum_{i\in M} \alpha_i x_i \geq 0$ *iff* $I^- \neq \emptyset$*, (iii)* $x_i \geq 0$ *iff* $(M\backslash\{i\}) \cap I^+ \neq \emptyset$ *or* $|M| = 1$*, (iv)* $x_i \leq 1$ *for* $i \in I^+$ *iff* $\sum_{j\in I^-} \alpha_j + \alpha_i \leq 0$*, (v)* $x_i \leq 1$ *for* $i \in I^-$ *iff* $\sum_{j\in I^+} \alpha_j + \alpha_i > 0$ *and (vi) (4.7) iff* $\sum_{i\in S^+} \alpha_i + \sum_{i\in I^-\backslash S^-} \alpha_i > 0$.

Inequality $y \leq 1$ is never facet-defining since it is dominated by the inequality obtained by setting $S^+ = I^+ \neq \emptyset$ and $S^- = I^-$ in (4.7). From a practical perspective, Theorem 4 is a negative result that states that nothing to gained over standard factorable relaxation techniques from exploiting a single complementarity constraint with side linear constraint as all its facet-defining inequalities are obtained through traditional linearization. This conclusion does not hold when several different linear constraints are considered as we illustrate next.

4.2 Convex Hull of $C^{1,2}$

We now illustrate how the cancel-and-relax procedure presented in Section 3 can be applied to obtain the convex hull of the one-complementarity set with two side constraints $C^{1,2}$. We motivate this result through the following example.

Example 4. Consider the set $C^{1,2}$ given by

$$C^{1,2} = \left\{ (x,y) \in [0,1]^5 \middle| \begin{array}{r} y(4x_1 + 7x_2 - 5x_3 - x_4) = 0 \\ 4x_1 + 7x_2 - 5x_3 - x_4 \geq 0 \\ -5x_1 - 3x_2 + 4x_3 + 6x_4 \geq 0 \end{array} \right\}.$$

The following is a facet-defining inequality of $\text{conv}(C^{1,2})$

$$20x_1 + 35x_2 - 25x_3 - 24x_4 + 23y \leq 23. \tag{4.8}$$

Theorem 3 implies that (4.8) can be obtained through a cancel-and-relax procedure involving cancellation of at most a single bilinear term, in this case $x_1 y$. Since the set does not satisfy Assumption 1, we introduce x_0 as discussed before Proposition 5. Then, we use the linear constraint to cancel the coefficient of $x_1 y$ in the bilinear constraint. More precisely, we aggregate the bilinear constraint $y(x_0 + 4x_1 + 7x_2 - 5x_3 - x_4) = 0$ and the linear constraint $x_0 - 5x_1 - 3x_2 + 4x_3 + 6x_4 \geq 0$ with weights -5 and $4(1-y)$ respectively to obtain

$$4x_0 - 9x_0 y - 20x_1 - 23x_2 y - 12x_2 + 9x_3 y + 16x_3 - 19x_4 y + 24x_4 \geq 0. \tag{4.9}$$

We now use McCormick relaxation to linearize (4.9). Formally, we aggregate (4.9), $x_0 \geq 0$, $-x_2 \geq -1$, $x_3 \geq 0$, and $x_4 \geq 0$ with weights $9y$, $23(1-y)$, $9(1-y)$ and $19y$ respectively and substitute $x_0 = 0$ to obtain (4.8).

The above procedure can be generalized to obtain a linear description of the convex hull of one-complementarity sets with two linear constraints:

$$C^{1,2} = \left\{ (x,y) \in [0,1]^{m+1} \;\middle|\; \begin{array}{l} y(\sum_{i\in M} \alpha_i x_i) = 0 \\ \sum_{i\in M} \alpha_i x_i \geq 0 \\ \sum_{i\in M} \beta_i x_i \geq 0 \end{array} \right\}.$$

First, we will select a variable x_l whose coefficient we will cancel. There are two cases. Either $\alpha_l > 0$ or $\alpha_l < 0$. In the first case, we multiply the bilinear constraint by β_l and the linear constraint $\sum_{i\in M} \beta_i x_i \geq 0$ by $\alpha_l(1-y)$, and sum up the resulting (in)equalities to obtain

$$-\sum_{i\in M} \alpha_l \beta_i x_i + \sum_{i\in M} \gamma_i x_i y \leq 0,$$

where we define $\gamma_i = \alpha_l\beta_i - \alpha_i\beta_l$ for $i \in M$. We now use McCormick inequalities to linearize the bilinear terms $x_i y$. This is done differently for terms for which γ_i is positive and negative and therefore we introduce $T^+ = \{i \in N \,|\, \gamma_i > 0\}$, and $T^- = \{i \in M \,|\, \gamma_i < 0\}$. We obtain

$$-\sum_{i\in M} \alpha_l \beta_i x_i + \sum_{i\in T^+} \gamma_i \min\{0, x_i + y - 1\} + \sum_{i\in T^-} \gamma_i \max\{x_i, y\} \leq 0,$$

which can be linearized in a way similar to that used in Section 4.1. The case where $\alpha_l < 0$ can be handled similarly, defining again $\gamma_i = \alpha_l\beta_i - \alpha_i\beta_l$ for $i \in M$, except that inequalities are reversed because weight $\alpha_l(1-y)$ is nonpositive. Proposition 10 then follows.

Proposition 10. *The convex hull of $C^{1,2}$ is given by $0 \leq x_i \leq 1$, $0 \leq y \leq 1$, $\sum_{i\in M} \alpha_i x_i \geq 0$, $\sum_{i\in M} \beta_i x_i \geq 0$, together with the linearization inequalities (4.7) and the following two families of 1-cancellation inequalities*

$$\sum_{i\in M} -\alpha_l\beta_i x_i + \sum_{i\in S^+} \gamma_i x_i + \sum_{j\in S^-} \gamma_j x_j + \left(\sum_{i\in S^+} \gamma_i + \sum_{j\in T^-\setminus S^-} \gamma_j \right) y \leq \sum_{i\in S^+} \gamma_i$$

for all $l \in M$ with $\alpha_l > 0$ and for all $S^+ \subseteq T^+, S^- \subseteq T^-$, as well as

$$\sum_{i\in M} \alpha_l\beta_i x_i + \sum_{i\in S^+} -\gamma_i x_i + \sum_{j\in S^-} -\gamma_j x_j + \left(-\sum_{i\in T^+\setminus S^+} \gamma_i - \sum_{j\in S^-} \gamma_j \right) y \leq -\sum_{j\in S^-} \gamma_j$$

for all $l \in M$ with $\alpha_l < 0$ and for all $S^+ \subseteq T^+, S^- \subseteq T^-$.

Note that the results of Proposition 10 can be extended in a straightforward fashion to problems with k side linear constraints through the above-discussed cancel-and-relax procedure. The major hurdle then becomes notational as we must consider all ways of canceling up to $k-1$ coefficients of the complementarity constraint using the additional linear constraints.

5 Conclusion

In this paper, we studied convexification methods for LPCCs, a class of models with many applications in engineering and economics. We showed that the use of RLT suggested in the literature does not convexify the problem but an extension with non-polynomial product-factors yields the desired sequential convexification properties. When the complementary variables do not appear in the linear constraints, traditional RLT applies. We showed that all nontrivial facets are obtained through a cancel-and-relax procedure. This procedure was used to obtain closed-form expressions for the convex hulls of one-complementarity sets with few side constraints. We also showed that convex hull characterization of one-cardinality sets can be used to generate facet-defining inequalities for sets with multiple constraints through lifting. This study provides a new set of tools for developing branch-and-cut approaches for solving LPCCs and more generally mathematical programs with complementarity constraints.

References

1. Balas, E.: Disjunctive programming: Properties of the convex hull of feasible points. Discrete Applied Mathematics 89(1-3), 3–44 (1998); original manuscript was published as a technical report in 1974
2. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0−1 programs. Mathematical Programming 58, 295–324 (1993)
3. de Farias, I.R., Johnson, E.L., Nemhauser, G.L.: Facets of the complementarity knapsack polytope. Mathematics of Operations Research 27, 210–226 (2002)
4. Ferris, M.C., Pang, J.S.: Engineering and economic applications of complementarity problems. SIAM Review 39, 669–713 (1997)
5. Hu, J., Mitchell, J.E., Pang, J.S., Yu, B.: On linear programs with linear complementarity constraints. Journal of Global Optimization (to appear)
6. Jeroslow, R.G.: Cutting-planes for complementarity constraints. SIAM Journal on Control and Optimization 16(1), 56–62 (1978)
7. Rockafellar, R.T.: Convex analysis. Princeton University Press, Princeton (1970)
8. Rockafellar, R.T., Wets, R.J.B.: Variational analysis. A Series of Comprehensive Studies in Mathematics. Springer, Berlin (1998)
9. Sherali, H.D., Adams, W.P.: A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM Journal on Discrete Mathematics 3, 411–430 (1990)
10. Sherali, H.D., Krishnamurthy, R.S., Al-Khayyal, F.A.: Enumeration approach for linear complementarity problems based on a reformulation-linearization technique. Journal of Optimization Theory and Applications 99, 481–507 (1998)
11. Tawarmalani, M.: Inclusion certificates and disjunctive programming. presented in Operations Research Seminar at GSIA, Carnegie Mellon University (2006)
12. Tawarmalani, M.: Inclusion certificates and simultaneous convexification of functions. Mathematical Programming (2010) (submitted)
13. Vandenbussche, D., Nemhauser, G.L.: A polyhedral study of nonconvex quadratic programs with box constraints. Mathematical Programming 102, 531–557 (2005)

Iterative Packing for Demand and Hypergraph Matching

Ojas Parekh

Sandia National Laboratories*, MS 1316,
Albuquerque NM 87185, USA
odparek@sandia.gov

Abstract. Iterative rounding has enjoyed tremendous success in elegantly resolving open questions regarding the approximability of problems dominated by covering constraints. Although iterative rounding methods have been applied to packing problems, no single method has emerged that matches the effectiveness and simplicity afforded by the covering case. We offer a simple iterative packing technique that retains features of Jain's seminal approach, including the property that the magnitude of the fractional value of the element rounded during each iteration has a direct impact on the approximation guarantee. We apply iterative packing to generalized matching problems including demand matching and k-column-sparse column-restricted packing (k-CS-PIP) and obtain approximation algorithms that essentially settle the integrality gap for these problems. We present a simple deterministic $2k$-approximation for k-CS-PIP, where an $8k$-approximation was the best deterministic algorithm previously known. The integrality gap in this case is at least $2(k-1+1/k)$. We also give a deterministic 3-approximation for a generalization of demand matching, settling its natural integrality gap.

1 Introduction

The (maximum weight) matching problem is cornerstone combinatorial optimization problem that has been studied for over 40 years. The problem is succinctly stated as seeking a maximum weight collection of non-intersecting edges in a weighted graph. Matching problems have enjoyed continual interest over the years and have been generalized in several orthogonal directions. The *k-hypergraph matching* problem, which is also known as *k-set packing*, seeks a maximum weight collection of non-intersecting hyperedges in a weighted hypergraph, with the additional restriction that each hyperedge contain at most k vertices. Thus 2-hypergraph matching is precisely the matching problem. While matching is in P, k-hypergraph matching is NP-complete for $k > 2$.

* Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 349–361, 2011.

Another direction in which matching has recently been generalized is through the augmentation of demands. The *demand matching* problem, introduced by Shepherd and Vetta [15], is defined on a weighted graph G that possesses a demand, $d_e \in \mathbb{Z}_+$ for each edge e, and a capacity $b_v \in \mathbb{Z}_+$ for each vertex v. We seek a maximum weight collection of edges M such that for any vertex v, the sum of the demands of the edges in M incident upon v is at most b_v (i.e. $\sum_{e \in M \cap \delta(v)} d_e \leq b_v$). Demand matching is a common generalization of the matching and knapsack problems: If $d_e = b_v = 1$ for all e and v, we recover matching, and by taking G to be a star we may model knapsack. Demand matching is MAXSNP-complete, even in the uniform weight case [15]. Demands are a powerful feature that allow for richer modeling; however, in general the demand version of combinatorial optimization problem can be significantly more difficult to approximate than its unit demand counterpart [6,5].

Problem definition. We consider the *k-hypergraph demand matching* (k-HDM) problem, which is the natural common generalization of k-hypergraph matching and demand matching. More formally, given a weighted hypergraph $H = (V, \mathcal{E})$ endowed with a demand $d_S \in \mathbb{Z}_+$ for each (hyper)edge $S \in \mathcal{E}$ and a capacity $b_v \in \mathbb{Z}_+$ for $v \in V$, the problem may be defined by the following integer program (k-HDM):

$$\begin{aligned} \text{Maximize} \quad & \sum_{S \in \mathcal{E}} c_S x_S \\ \text{subject to} \quad & \sum_{S | v \in S} d_S x_S \leq b_v \qquad \forall v \in V \\ & x_S \in \{0,1\} \quad \forall S \in \mathcal{E} \ , \end{aligned}$$

where $|S| \leq k$ for each edge $S \in \mathcal{E}$. Note the latter restriction yields a constraint matrix with at most k nonzeros per column. This problem is also known as the *k-column-sparse column-restricted packing integer program* (k-CS-CPIP) problem. It is a specialization of the *k-column-sparse packing integer program* (k-CS-PIP) problem, in which we allow each $S \in \mathcal{E}$ to have different demand values d_v^S at each vertex v.

We make the assumption that for each edge S, $d_S \leq b_v$ for all $v \in S$. This so-called *no-clipping* assumption is easy to satisfy by deleting edges that violate it; however, this assumption is necessary in order for the natural LP relaxation to have a bounded integrality gap. We note that the restriction that $x_S \in \{0,1\}$ is for the sake of exposition and that our results may be extended to apply to multiple copies of edges.

Results. Singh and Lau [16] were the first to extend Jain's celebrated iterative rounding technique [9] to address packing constraints. Their approach obtains an approximate solution that marginally violates the packing constraints by iteratively removing packing constraints involving only a small number of variables. They were able to apply this elegant idea to resolve an open question concerning minimum cost degree-bounded spanning trees. More recently, Chan

and Lau [4] employed an interesting combination of an iterative approach and the fractional local ratio method [2] to give the first approximation algorithm for the k-hypergraph matching problem that matched the integrality gap of the natural LP formulation, which had previousy been established as $k-1+1/k$ [7].

Our main insight, which differentiates our approach from previous ones, is to iteratively maintain a sparse approximate convex decomposition of the current fractional solution. This affords us a simple pseudo-greedy technique called iterative packing that yields improved approximation algorithms for k-HDM (k-CS-CPIP) and special cases that essentially settle the integrality gap of the natural LP formulations. For instance, iterative packing is able to establish an integrality gap of $k-1+1/k$ for not just k-hypergraph matching but for k-hypergraph b-matching as well.

Akin to Jain's iterative rounding method for covering problems [9], iterative packing is able to leverage large fractional edges to obtain stronger approximation guarantees. As mentioned above, iterative packing produces and maintains a sparse approximate convex decomposition rather than a single solution, which is likely to have additional applications. We motivate the technique on the standard matching problem in the next section.

Our first result is a deterministic $2k$-approximation for k-HDM (k-CS-CPIP) based on the natural LP relaxation. The integrality gap of this relaxation is at least $2(k-1+1/k)$ (see Sect. 3), hence our result essentially closes the gap. Prior to our work, deterministic $8k$-approximations [5,1] and a randomized $(ek+o(k))$-approximation [1] were the best known. Moreover, even the special case of k-hypergraph matching cannot be approximated within a factor of $\Omega(\frac{k}{\log k})$ unless P=NP [8].

With a more refined application of iterative packing, we are able to derive a 3-approximation for 2-CS-PIP, which generalizes the demand matching problem. Prior to our work, a deterministic 3.5-approximation and randomized 3.264-approximation for demand matching were given by Shepherd and Vetta [15]. Chakrabarty and Pritchard [14] recently gave a deterministic 4-approximation and randomized 3.764-approximation for 2-CS-PIP. Shepherd and Vetta also established a lower bound of 3 on the integrality gap of the natural LP for demand matching, hence our result settles the integrality gap for both 2-CS-PIP and demand matching at 3.

Related work. Chekuri, Mydlarz, and Shepherd [6] presented an approximation algorithm with $O(k)$ guarantee for the restricted version of k-HDM in which $\max_S d_S \leq \min_v b_v$. Their result is part of a framework which they developed based on work of Kolliopoulos and Stein [10] that relates the integrality gap of a demand-endowed packing problem to its unit demand counterpart.

While Chekuri *et al.* [5] observed an $8k$-approximation for k-HDM, a recent flurry of work has also yielded $O(k)$-approximations for the more general k-CS-PIP problem. Pritchard initiated the improvements with an iterative rounding based $2^k k^2$-approximation [13], which was improved to an $O(k^2)$-approximation

by Chekuri, Ene, and Korula (see [14] and [1]) and Chakrabarty and Pritchard [14]. Most recently, Bansal *et al.* [1] devised a deterministic $8k$-approximation and a randomized $(ek + o(k))$-approximation.

Outline. In the following section we motivate iterative packing with an example. In Sect. 3 we apply the method to design a $2k$-approximation for k-HDM. We finally present a more refined application in Sect. 4 to derive a 3-approximation for 2-CS-PIP.

2 Iterative Packing: An Example

We illustrate iterative packing on the maximum matching problem. Although this is a simple application, it serves well to illustrate the method. Consider the natural degree-based LP relaxation $P_M(G)$ for the maximum matching problem on a graph $G = (V, E)$:

$$\begin{aligned} \text{Maximize} \quad & \sum_{e \in E} c_e x_e && P_M(G) \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e \leq 1 \quad \forall v \in V && (1) \\ & 0 \leq x_e \leq 1 \quad \forall e \in E. && (2) \end{aligned}$$

Given a feasible fractional solution $x^* \in P_M(G)$, the iterative packing procedure obtains an α-*approximate* convex decomposition of x^*,

$$\alpha x^* = \sum_{i \in \mathcal{I}} \lambda_i \chi^i \ , \tag{3}$$

for some $\alpha \in (0, 1]$, where each $\chi^i \in P_M(G)$ is an integral solution (and $\sum_i \lambda_i = 1$ and $\lambda_i \geq 0$ for all i). Iterative packing in its most basic form directly produces a sparse decomposition, namely one with $|\mathcal{I}| \leq |E| + 1$. Even when this is not the case, we can apply elementary linear algebra to retain at most $|E| + 1$ solutions (more generally $n + 1$, where $x^* \in \mathbb{R}^n$). A procedure to accomplish the latter is related to Carathéodory's Theorem and makes for a good exercise.

The construction of the decomposition (3) implies that one can find an integral solution with cost at least $\alpha(cx^*)$, thus $1/\alpha$ corresponds to the approximation guarantee of the resulting approximation algorithm. A nice feature is that the decomposition gives us a cost oblivious representation of an approximate solution.

For $P_M(G)$, we first show that choosing $\alpha = 1/2$ suffices. This yields a 2-approximation while also showing that the integrality gap of $P_M(G)$ is at most 2. We then show that we may select $\alpha = 2/3$ by leveraging the fact that extreme points of $P_M(G)$ must contain an edge e with $x_e \geq 1/2$ (in fact this holds for all e). The latter precisely matches the integrality gap of $P_M(G)$. This is interesting, since much like iterative rounding, iterative packing offers insight into how large

fractional components can facilitate the approximation of packing problems. Akin to iterative rounding, iterative packing is motivated by a simple idea. We start with a fractional solution x^* and:

1. Remove an edge e (without otherwise modifying the instance)
2. Recursively obtain an α-approximate convex decomposition of the resulting fractional solution, $\bar{x}^*$
3. Pack e into precisely an αx_e^* fraction of the integral solutions.

The key, of course, is showing that the last step can always be performed successfully. For this to work, we require that for any fractional (or perhaps extreme point) solution x^* there exists an $e \in E$ with

$$\sum_{i \in \mathcal{I}_e} \lambda_i \geq \alpha x_e^* \ , \tag{4}$$

where $\alpha \bar{x}^* = \sum_{i \in \mathcal{I}} \lambda_i \chi^i$ is an arbitrary approximate convex decomposition of the residual solution, $\bar{x}^*$, and $i \in \mathcal{I}_e$ indicates that χ^i is able to accommodate the edge e (i.e. $\chi^i \cup e$ is a matching).

Although we may well be able to pack e into a fraction of the integral solutions that is larger than an αx_e^*, to maintain our implicit inductive hypothesis we must ensure that e is packed into exactly an αx_e^* fraction of solutions. To accomplish this, we may have to clone some solution χ^i, insert e into exactly one of the two copies of χ^i, and distribute the multiplier λ_i among the copies so that e appears in the requisite fraction of solutions. The base case, which contains no edges, selects the empty solution with a multiplier of 1. Thus if (4) holds universally for a particular value of α, then we can efficiently obtain an α-approximate convex decomposition of x^* consisting of at most $|E| + 1$ integral solutions. Selecting the best of these gives us the corresponding approximation algorithm.

To see that (4) holds when $\alpha = 1/2$, consider some fractional solution x^* and an arbitrary edge $e = uv \in E$ with $x_e^* > 0$. Obtaining $\bar{x}^*$ as above by deleting e, we have that

$$\max\{\bar{x}^*(\delta(u)), \bar{x}^*(\delta(v))\} \leq 1 - x_e^* \ ,$$

hence in any convex decomposition $\alpha \bar{x}^*$, at most a $2\alpha(1 - x_e^*)$ fraction of the χ^i do not accomodate e, hence we require $1 - 2\alpha(1 - x_e^*) \geq \alpha {x^*}_e$, which is equivalent to

$$\alpha \leq \frac{1}{2 - x_e^*} \tag{5}$$

Thus by selecting $\alpha = 1/2$, we may successfully pack any edge $0 \leq x_e^* \leq 1$ in the last step of our algorithm. However, by selecting a large edge at each iteration we can improve the bound. It is well known that extreme points of $P_M(G)$ are 1/2-integral, so we may actually take $\alpha = 1/(2 - 1/2) = 2/3$. More generally – just as with iterative rounding – it suffices to show that an extreme point always contains some edge of large fractional value. We explore this idea in conjunction with 2-CS-PIP in Sect. 4. However, in the next section we show that the framework above with a simple modification yields a $2k$-approximation for k-HDM.

3 Iterative Packing for *k*-Hypergraph Demand Matching

The results in this section are obtained using the framework outlined for our the matching problem in the previous section:

1. Remove a (hyper)edge S (without otherwise modifying the instance)
2. Recursively obtain an α-approximate convex decomposition of the resulting fractional solution, $\bar{x}^*$
3. Pack e into precisely an αx^*_S fraction of the integral solutions.

However, directly applying the algorithm above to k-HDM does not yield a bounded approximation guarantee (w.r.t. k). We show that simply selecting an edge S with minimal demand in step 1 above yields a $2k$-approximation.

As with our analysis in the previous section, the crux lies in being able to carry out step 3 successfully. Following the analysis in Sect. 2, let $\alpha\bar{x}^* = \sum_{i\in\mathcal{I}} \mu_i \chi^i$ be an arbitrary convex decomposition of the residual solution $\bar{x}^*$ obtained in step 2. We may inductively assume the existence of such a decomposition (with a trivial base case). To determine whether the edge S may be packed in the requisite αx^*_S fraction of the integral solutions χ^i, we consider the potential fraction of solutions in which S is blocked at each $u \in S$. For $u \in S$, let β_u be the fraction of solutions in which S cannot be packed at u. We may think of β_u as the fraction of bad solutions in terms of packing S. In the worst case, S is blocked pairwise disjointly at each incident vertex.

Lemma 1. *Edge S may be packed into an αx^*_S fraction of the integral solutions χ^i, provided*

$$1 - \sum_{u\in E} \beta_u \geq \alpha x^*_S \ .$$

Proof. The lemma follows from the same reasoning used to derive a union bound if x^* were a probability distribution. The quantity $\sum_{u\in S} \beta_u$ represents the maximum fraction of solutions in which S is blocked at some incident vertex, hence $1 - \sum_{u\in S} \beta_u$ is the minimum fraction of solutions into which it is feasible to insert S. □

We may derive a $1/\alpha$-approximation guarantee on the performance of iterative packing by bounding β_u and selecting α so that Lemma 1 is satisfied. For this purpose we find it useful to think of the residual convex decomposition, $\alpha\bar{x}^* = \sum_{i\in\mathcal{I}} \mu_i \chi^i$, obtained in step 2. above, as inducing a collection of bins at each $u \in V$ where each bin has capacity b_u. Each $i \in \mathcal{I}$ induces a bin of width μ_i; the height h_i is equal to the sum of the demands of the edges incident upon u that appear in the solution χ^i; that is

$$h_i := \sum_{S\in\chi^i | u\in S} d_S \ .$$

Thus the aggregate capacity of the bins is at most $(\sum_{i\in\mathcal{I}} \mu_i)b_u = b_u$, where each bin contains a volume of $\mu_i h_i$.

Next we bound the fraction of bad solutions at u. For this, we define $\bar{\delta} = \min_{T\in\mathcal{E}|T\neq S} d_T$, i.e. $\bar{\delta}$ is the minimal demand in the residual instance.

Lemma 2. *For the convex decomposition,* $\alpha\bar{x}^* = \sum_{i\in\mathcal{I}} \mu_i \chi^i$*, we have*

$$\beta_u \leq \alpha \frac{b_u - d_S x^*_S}{\max\{b_u - d_S + 1,\, \bar{\delta}\}} \ .$$

Proof. Let $\mathcal{I}_{\overline{S}}$ be the indices of the bins which cannot accommodate S. Thus by definition,

$$\beta_u = \sum_{i\in\mathcal{I}_{\overline{S}}} \mu_i \ .$$

The total volume of all such bins is at most the total u-volume of $\alpha\bar{x}$, which does not contain S:

$$\sum_{i\in\mathcal{I}_{\overline{S}}} \mu_i h_i \leq \alpha(b_u - d_S x^*_S) \ .$$

Each bin in $\mathcal{I}_{\overline{S}}$ must have height large enough to block S and must also contain at least one edge since S fits on its own, by the no clipping assumption. Thus $h_i \geq \max\{b_u - d_S + 1,\, \bar{\delta}\}$, yielding the desired result when coupled with the above equation and inequality:

$$\max\{b_u - d_S + 1,\, \bar{\delta}\}\beta_u = \max\{b_u - d_S + 1,\, \bar{\delta}\} \sum_{i\in\mathcal{I}_{\overline{S}}} \mu_i \leq \alpha(b_u - d_S x^*_S) \ . \qquad \square$$

Unfortunately when d_S is large and x^*_S is small, the above bound may be large. However, by appealing to special cases of k-HDM or by more carefully selecting the edge S, the bound β_u becomes manageable. For instance, consider the case of the k-hypergraph b-matching problem, obtained when $d_S = 1$ for all S. In this case $\beta_u \leq \alpha$ by Lemma 2, which allows us to satisfy the hypothesis of Lemma 1 by selecting α such that:

$$\alpha \leq \frac{1}{x^*_S + k} \Rightarrow \alpha x^*_S \leq 1 - k\alpha \leq 1 - \sum_{u\in E} \beta_u \ . \tag{6}$$

Since $x^*_S \leq 1$ for all E, we may universally select $\alpha = \frac{1}{k+1}$ for all S, yielding an approximation guarantee of $k + 1$. Krysta [12] proved that a greedy algorithm also achieves this bound, and Young and Koufogiannakis [11] give a primal dual algorithm achieving a bound of k, which is the best known. Although we omit the details in this article, iterative packing can be used to show that the integrality gap of the natural LP relaxation for k-hypergraph b-matching is at most $k - 1 + 1/k$, which settles the gap.

Turning our attention back to k-HDM, the "max" in Lemma 2's bound hints at our strategy: we shall always select an edge S with minimal demand, so that $\bar{\delta}$ is large enough to be of value. In fact the resulting approximation algorithm applies to a generalization of k-HDM (k-CS-CPIP) in which we allow each edge S to have a different valued demand, d^S_v at each vertex v, as is allowed in k-CS-PIP. However, we require that the edges can be ordered, $S_1, S_2, \ldots, S_m$, so that for any distinct S_i, S_j with $u \in S_i \cap S_j$, we have $d^{S_i}_u \leq d^{S_j}_u$ if $i \leq j$; that is, the demands monotonically increase at every vertex. Note that this is clearly

the case with k-HDM, where $d_S = d_u^S = d_v^S$ for all $u, v \in S$. We may simply sort the demands over the edges to obtain such an ordering. We perform iterative packing with such an ordering (i.e. select an S with minimal demand). Now when we insert S back into the approximate convex decomposition of $\alpha\bar{x}$, we may assume that $d_S \leq \bar{\delta}$.

Theorem 1. *Iterative packing applied to k-CS CPIP with the edges inserted in order of nonincreasing demand is a $2k$-approximation.*

Proof. To simplify our analysis, we conduct it in terms of $1/\alpha$. The stipulation of Lemma 1 may be expressed as:

$$x_S^* + \sum_{u \in S} \frac{\beta_u}{\alpha} \leq \frac{1}{\alpha} \ .$$

Our goal is to show that the above holds when $1/\alpha = 2k$. By applying the bound on β_u from Lemma 2, we reduce our task to showing that

$$x_S^* + \sum_{u \in S} \frac{b_u - d_S x_S^*}{\max\{b_u - d_S + 1,\, \bar{\delta}\}} \leq 2k \ ,$$

for any value of $x_S^* \in [0, 1]$. Note that when the left hand side above is considered as a function of the parameters b_u, d_S, and x_S^*, it is linear in x_S^*. Thus it is maximized in one of the cases $x_S^* = 0$ or $x_S^* = 1$. When $x_S^* = 1$ we indeed have

$$1 + \sum_{u \in S} \frac{b_u - d_S}{\max\{b_u - d_S + 1,\, \bar{\delta}\}} \leq 1 + \sum_{u \in S} \frac{b_u - d_S}{b_u - d_S + 1} \leq 1 + k \leq 2k \ .$$

On the other hand when $x_S^* = 0$, we have

$$0 + \sum_{u \in S} \frac{b_u}{\max\{b_u - d_S + 1,\, \bar{\delta}\}} \leq 2 \sum_{u \in S} \frac{b_u}{b_u + \bar{\delta} - d_S + 1} \leq 2k \ ,$$

where the former inequality follows because $\max\{x, y\} \geq (x + y)/2$, and the latter holds because our ordering of the edges gives us $\bar{\delta} \geq d_S$. □

Integrality gap. Our result essentially settles the integrality gap of the natural formulation. As noted in [7] the projective plane of order $k-1$ yields an integrality gap of at least $k - 1 + 1/k$, even for the case of k-hypergraph matching. For k-HDM (and consequently k-CS PIP) one may obtain a lower bound approaching $2(k - 1 + 1/k)$ by again considering a projective plane of order $k - 1$, setting all the demands to d and each capacity to $b = 2d - 1$.

4 Improvements for 2-CS-PIP and Demand Matching

Here we consider the general k-CS-PIP rather than k-HDM/k-CS-CPIP, but for the special case $k = 2$. This case is of particular interest as it is natural

generalization of the demand matching problem (i.e. 2-CS-CPIP), which itself is combination of both b-matching and knapsack type problems in graphs.

Shepherd and Vetta [15] were able to show that integrality gap of the natural LP formulation was between 3 and 3.264; however, establishing the exact value has remained an open problem prior to our work. We are able to show that there is indeed a 3-approximation based on the natural LP for not only demand matching but also the more general 2-CS-PIP problem. This consequently settles the integrality gaps for both problems. Although designing a polynomial time algorithm takes a bit of work, iterative packing allows us to establish the integrality gap relatively easily.

4.1 Establishing the Integrality Gap

As observed in the introduction for the standard matching problem, iterative packing readily yields an upper bound of 2 on the integrality gap of the natural formulation, which is sharpened to the optimal value of $3/2$ $(= k - 1 + 1/k)$ by observing that extreme points must contain some large component ($x_e \geq 1/2$) and iterating only on such edges. For 2-CS PIP we also apply iterative packing solely on large components in extreme points – in fact, those with $x_e = 1$ when they exist.

An interesting phenomenon with general demand packing problems is that 1-edges (i.e. $x_e = 1$) cannot simply swept under the rug as with $\{0, 1\}$-demand problems. For the latter, one can simply assume such edges are selected in a solution and obtain a straightforward residual instance and feasible fractional solution. With general demand problems, however, such an approach may violate the no clipping assumption. Iterative packing, on the other hand, performs quite well on 1-edges, which allows us to rather easily prove an optimal integrality gap. Although we will also derive this result in the next section by means of a polynomial time algorithm, in this section we attack only the integrality gap with a short proof that highlights the effectiveness of iterative packing when augmented with simple insights about the nature of extreme points. Our first task is to show that extreme points without 1-edges admit a manageable structure. We note that since our discussion involves the more general 2-CS-PIP rather than demand matching, each edge uv may have distinct demands d_u^{uv} and d_v^{uv}.

Lemma 3. *If $\hat{x}$ is an extreme point of the natural LP then the fractional part of $\hat{x}$ induces connected components with at most one cycle. If we have $0 < \hat{x} < 1$, then $\hat{x}$ induces vertex disjoint cycles.*

Proof. Let $F \subseteq E$ be the fractional support of $\hat{x}$. For each connected component C induced by F, the only tight constraint in which $e \in F(C)$ may appear are degree constraints,

$$\forall u \in V : \sum_{uv \in \delta(u)} d_u^{uv} x_{uv} \leq b_u \ ,$$

for $u \in V(C)$. Thus $|F(C)| \leq |V(C)|$, otherwise we would find that a basis for $\hat{x}$ contains linearly dependent columns among those of $F(C)$. This establishes the first claim of the lemma.

If $0 < \hat{x} < 1$, then no component induced by $\hat{x}$ may contain an edge e incident to a leaf l, otherwise we would have $d_l^e \hat{x}_e = b_l$, implying $\hat{x}_e = 1$ by the no clipping assumption. Thus we must have $|F(C)| = |V(C)|$ for each component C, and since we have no leaves, C must be a cycle. □

Coupled with our earlier analysis of iterative packing, this is the only fact we need to establish the integrality gap of 3. Consider the following non-efficient extension of iterative packing:

1. If x^* is not an extreme point, obtain a convex decomposition into extreme points, $x^* = \sum_i \mu_i \hat{x}^i$, and apply the algorithm to each extreme point $\hat{x}^i$.
2. If the extreme point $\hat{x}$ contains an integral edge let e be such an edge, otherwise let e be any edge.
3. Delete e to obtain $\bar{x}$ and recursively construct an approximate convex decomposition into integral solutions, $\frac{1}{3}\bar{x} = \sum_j \lambda_j \chi^j$.
4. Insert e into exactly a $\frac{1}{3}x_e$ fraction of the solutions χ^j.

Lemma 4. *Step 4. above can always be completed successfully.*

Proof. Suppose there is an integral $\hat{x}_e$ and that $\hat{x}_e = 1$ ($\hat{x}_e = 0$ clearly works). Substituting $\hat{x}_e = 1$ into the bound from Lemma 2 yields $\beta_u \leq \alpha$ for $u \in e$, which we have already observed (see (6)) allows us to select $\alpha = 1/(k+1) = 1/3$. On the other hand, if there is no integral edge by Lemma 3, $\hat{x}$ induces a 2-regular graph. When we delete e in this case, at each endpoint $u \in e$ there is a single edge, f_u remaining. Thus $\beta_u \leq \alpha \hat{x}_{f_u} \leq \alpha$ in this case as well. □

We have a lower bound on the integrality gap of $2(k - 1 + 1/k) = 3$ from the previous section. To complete our proof, we note that by assuming an approximate convex decomposition for each extreme point, $\frac{1}{3}\hat{x}^i = \sum_j \lambda_j \chi^{ij}$, we may obtain a decomposition for $\frac{1}{3}x^*$ as $\sum_i \sum_j \mu_i \lambda_j \chi^{ij}$.

Theorem 2. *The integrality gap of the natural LP formulation for 2-CS PIP is 3.*

4.2 A Polynomial Time Implementation

Unfortunately we do not currently have a means of directly implementing the algorithm above in polynomial time; however, Shepherd and Vetta [15] analyze the fractional structure of extreme points of the natural LP for demand matching. We are able to develop a polynomial time algorithm by relying on generalizations of their demand matching insights. A key ingredient used by Shepherd and Vetta is the augmentation of a fractional path (Sect. 4.1 in [15]). We begin by giving a generalization of this tool for the case of 2-CS PIP.

Lemma 5. *Let $P = (v_0, e_1, v_1, e_2, \ldots, v_k)$ be a path; there exists an augmentation vector $z \in \mathbb{R}^E$ such that*

1. $\sum_{uv \in \delta(u)} d_u^{uv} z_{uv} \neq 0$, *for* $u = v_0, v_k$
2. $\sum_{uv \in \delta(u)} d_u^{uv} z_{uv} = 0$, *for all other* $u \in V$

Proof. We set $z_e = 0$ for all $e \notin E(P)$, and we set $z_{e_1} = 1$. We now set the value of $z_{e_{i+1}}$, for $i \geq 1$, based on the value of z_{e_i} as follows:

$$z_{e_{i+1}} = -(d_{v_i}^{e_i} / d_{v_i}^{e_{i+1}}) z_{e_i} \quad .$$

The first condition is satisfied since $z_{e_i} \neq 0 \Rightarrow z_{e_{i+1}} \neq 0$, and the second condition holds since $d_{v_i}^{e_{i+1}} z_{e_{i+1}} = -d_{v_i}^{e_i} z_{e_i}$. □

Algorithm. We will explain the utility of the above lemma in just a moment; however, first we give an overview of our algorithm:

1. Find an extreme point $\hat{x}$ of the natural 2-CS PIP LP.
2. Delete any 0-edges and iterate on the 1-edges until a complete fractional solution $\bar{x}$ remains.
3. We will show that $\bar{x}$ possesses a structure that allows us to infer a 3-approximation algorithm based on a result of Shepherd and Vetta.
4. Apply a result of Carr and Vempala [3] with the above 3-approximation algorithm as an approximate separation oracle to obtain an approximate convex decomposition of $\bar{x}$ in polynomial time.
5. Pack the removed 1-edges into 1/3 of the solutions from the above decomposition.

The basic idea is to use a more complex base case for iterative packing, rather than the default case of an empty graph. In the above algorithm steps 3 and 4 represent the base case. We address the results employed in these steps in turn.

Analysis. First we describe the 3-approximation derived from Shepherd and Vetta's work. Note that in step 3, we have a solution $\bar{x}$ that contains precisely the fractional components of an extreme point $\hat{x}$. By Lemma 3, each component induced by $\bar{x}$ is either a tree or unicyclic. For the former case, we can apply a generalization of Thereom 4.1 from Shepherd and Vetta [15], which yields a 2-approximation with respect to a fractional solution x^* whose support is a tree. They use path augmentations to derive this result for demand matching, for which we give an appropriate generalization in Lemma 5.

A 3-approximation. We briefly explain the basic idea behind the 2-approximation mentioned above and recommend the reader consult Sect. 4.1 in [15] for a rigorous proof. Since x^* induces a tree, we can find a path P between two leaves s and t. We apply Lemma 5 on P to obtain z; we are able to select an $\varepsilon \neq 0$ such that:

- $x^* + \varepsilon z$ earns cost at least that of x^*
- $x^* + \varepsilon z$ is still feasible at every vertex except possibly s and t
- $x^* + \varepsilon z$ has some integral edge

Thus we obtain a new solution of no worse cost, but it may be infeasible at s and t. We temporarily remove any integral edges to obtain smaller trees and continue the procedure to obtain a collection of integral edges that are not necessarily feasible but are of superoptimal cost. The key observation is that when a vertex becomes infeasible, it is a leaf, which allows Shepherd and Vetta to conclude in the final solution, at every vertex there is at most one edge whose removal results in a feasible solution. Since the edges form a forest, Shepherd and Vetta are able to partition the edges into two sets such that each is feasible, yielding a 2-approximation.

Returning to our algorithm, each component of $\hat{x}$ contains at most one cycle, thus for a given cost function, we may delete the cheapest edge from each such cycle to retain a solution of cost at least $2/3$ that of $\hat{x}$. This leaves a forest on which we apply Shepherd and Vetta's procedure to obtain an integral solution of cost at least $1/3 = 1/2 \cdot 2/3$ that of $\hat{x}$. The trouble is that although this gives a 3-approximation, we actually need a convex decomposition of $1/3\hat{x}$ in order to pack the 1-edges removed in the second step. Luckily, we are able to appeal to the following result Carr and Vempala [3] to obtain such a decmoposition.

Theorem 3. *(Thm 2, [3]) Given an LP relaxation P, an r-approximation heuristic A, and any solution x^* of P, there is a polytime algorithm that finds a polynomial number of integral solutions $z^1, z^2, \ldots$ of P such that*

$$\frac{1}{r}x^* \leq \sum_j \lambda_j \chi^{z^j}$$

where $\lambda_j \geq 0$ for all j, and $\sum_j \lambda_j = 1$.

The theorem from Carr and Vempala is for covering problems; however, their ideas yield the result for the packing case as well. We also note that the heuristic A must be an r-approximation with respect to the lower bound given by the relaxation P.

Obtaining an exact decomposition. The only remaining detail is that for our purposes, we require an exact decomposition of x^*/r (i.e. $x^*/r = \sum_j \lambda_j \chi^{z^j}$). We observe that this can be done by at most doubling the number integral solutions z^j. For any edge e such that $x^*_e/r > \sum_j \lambda_j \chi^{z^j}_e$, we remove e from solutions that contain e. We continue this until removing e from any solution would give us $x^*_e/r \leq \sum_j \lambda_j \chi^{z^j}_e$. Now we clone some solution z^j that contains e and remove e from the clone; we distribute λ_j between these two solutions to attain $x^*_e/r = \sum_j \lambda_j \chi^{z^j}_e$. Finally, Lemma 4 shows that step 5. may be completed.

Carr and Vempala's result is essentially an application of LP duality and the polynomial time equivalence of separation and optimization. It certainly seems likely that a more direct analysis of $1/3\hat{x}$ could be used to construct a convex decomposition; however, we are enamored with the elegance of Carr and Vempala's result.

5 Concluding Remarks

We have obtained simple iterative packing algorithms for k-hypergraph demand matching problems that essentially settle the gap of the respective natural LP relaxations. Obvious open questions include generalizing our work to k-CS-PIP. We are currently investigating this and have derived promising partial results. It is conceivable that one may also develop an analogue of iterative packing for covering that appeals to approximate convex decompositions.

References

1. Bansal, N., Korula, N., Nagarajan, V., Srinivasan, A.: On k-column sparse packing programs. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 369–382. Springer, Heidelberg (2010)
2. Bar-Yehuda, R., Halldórsson, M.M., Naor, J., Shachnai, H., Shapira, I.: Scheduling split intervals. SIAM J. Comput. 36(1), 1–15 (2006)
3. Carr, R.D., Vempala, S.: Randomized metarounding. Random Struct. Algorithms 20(3), 343–352 (2002)
4. Chan, Y.H., Lau, L.C.: On linear and semidefinite programming relaxations for hypergraph matching. In: Charikar, M. (ed.) SODA, pp. 1500–1511. SIAM, Philadelphia (2010)
5. Chekuri, C., Ene, A., Korula, N.: Unsplittable flow in paths and trees and column-restricted packing integer programs. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX 2009. LNCS, vol. 5687, pp. 42–55. Springer, Heidelberg (2009)
6. Chekuri, C., Mydlarz, M., Shepherd, F.B.: Multicommodity demand flow in a tree and packing integer programs. ACM Transactions on Algorithms 3(3) (2007)
7. Füredi, Z., Kahn, J., Seymour, P.D.: On the fractional matching polytope of a hypergraph. Combinatorica 13(2), 167–180 (1993)
8. Hazan, E., Safra, S., Schwartz, O.: On the complexity of approximating -set packing. Computational Complexity 15(1), 20–39 (2006)
9. Jain, K.: A factor 2 approximation algorithm for the generalized Steiner network problem. Combinatorica 21, 39–60 (2001)
10. Kolliopoulos, S.G., Stein, C.: Approximating disjoint-path problems using packing integer programs. Mathematical Programming 99(1), 63–87 (2004)
11. Koufogiannakis, C., Young, N.E.: Distributed fractional packing and maximum weighted b-matching via tail-recursive duality. In: Keidar, I. (ed.) DISC 2009. LNCS, vol. 5805, pp. 221–238. Springer, Heidelberg (2009)
12. Krysta, P.: Greedy approximation via duality for packing, combinatorial auctions and routing. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 615–627. Springer, Heidelberg (2005)
13. Pritchard, D.: Approximability of sparse integer programs. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 83–94. Springer, Heidelberg (2009)
14. Pritchard, D., Chakrabarty, D.: Approximability of sparse integer programs (2010) (to appear in Algorithmica) 19 p.
15. Shepherd, F.B., Vetta, A.: The demand-matching problem. Mathematics of Operations Research 32(3), 563–578 (2007)
16. Singh, M., Lau, L.C.: Approximating minimum bounded degree spanning trees to within one of optimal. In: Johnson, D.S., Feige, U. (eds.) STOC, pp. 661–670. ACM, New York (2007)

Universal Packet Routing with Arbitrary Bandwidths and Transit Times

Britta Peis* and Andreas Wiese**

Technische Universität Berlin, Straße des 17. Juni 136,
10623 Berlin, Germany
{peis,wiese}@math.tu-berlin.de

Abstract. We prove bounds for the length of optimal schedules for store-and-forward packet routing in the setting of arbitrary bandwidths and transit times. The problem is commonly studied only in the setting of unit bandwidths and unit transit times. Our results generalize the existing work to a much broader class of instances and also improve the known bounds significantly. For the case of unit transit times and bandwidths we improve the best known bound of $39(C + D)$ to $23.4(C + D)$, where C and D denote the trivial lower bounds congestion and dilation. If every link in the network has a certain minimum transit time or capacity our bounds improve even further up to $4.32(C + D)$. Key to our results is a framework which employs tight bounds for instances where each packet travels along only a small number of edges. Further insights for such instances would reduce our constants even more. This is the first improvement of the bounds for this very fundamental problem in more than 10 years.

1 Introduction

The problem to transport packets within a communication network on time is one of the most fundamental problems in parallel or distributed systems. Routing protocols need to be designed that find a path for each individual packet along which it is routed through the network. Once the paths are chosen, the protocol needs to determine a schedule which defines when each packet traverses the communication links of its path. In this paper we study *universal* routing protocols [24] for the latter task. They are universal in the sense that they work for *arbitrary* networks, as well as for *arbitrary* paths (e.g., the predefined paths need not necessarily be shortest paths).

Most parallel and distributed systems utilize *store-and-forward packet routing* in which no two packets can cross the same link simultaneously ("unit bandwidth"), and each packet needs one unit of time to traverse a single link ("unit transit times"). Usually, the performance of a routing protocol is measured in

* Supported by DFG grant PE 1434/3-1.
** Supported by the DFG Focus Program 1307 within the project "Algorithm Engineering for Real-time Scheduling and Routing".

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 362–375, 2011.

terms of the two trivial lower bounds C and D. The *congestion* C denotes the maximal number of paths using a single link. The *dilation* D denotes the maximal length of a path along which a packet has to be routed.

The break-through result for store-and-forward packet routing is certainly due to Leighton, Maggs and Rao [12] that proves the existence of a routing protocol whose length is linear in $C+D$. However, the hidden constants in their result are very large. Scheideler [24] improves this by showing the existence of a protocol of length at most $39(C + D)$.

In contrast to previous work, we provide bounds for the more general setting of store-and-forward packet routing with arbitrary transit times and bandwidths. For the special case of unit transit times and bandwidths, we can even improve the so far best known result of Scheideler from $39(C + D)$ to $23.4(C + D)$.

Note that the restriction to unit transit times and bandwidths in ordinary store-and-forward packet routing is of course justifyable as one could always transform an instance with arbitrary transit times and bandwidths to the latter by splitting the edges and adding parallel edges, respectively. However, as we will show in this paper, the bounds on the length of an optimal protocol improve considerably for increasing minimum bandwidth or increasing minimum transit times (or both) of the communication links, see Table 1. For example, if we know that the bandwidth is at least two on each link, the bound decreases already to $21.59(C + D)$, even if we have unit transit times for all edges. Note that in the case of arbitrary bandwidths the (generalized) congestion C depends on the bandwidths of the edges. In particular, the improvement of the bound is *not* a trivial consequence of the increased bandwidth.

1.1 The Model

We model the communication network by a directed graph $G = (V, E)$ whose edges correspond to the links of the network. Each edge $e \in E$ is equipped with a certain bandwidth $b_e \in \mathbb{N}$ denoting the maximal number of packets that are allowed to traverse e simultaneously, and a certain transit time $\tau_e \in \mathbb{N}$ denoting the time needed for a single packet to traverse e. We define $b := \min_{e \in E} b_e$ and $\tau := \min_{e \in E} \tau_e$.

Each packet M_i must be sent through the network from its origin $s_i \in V$ to its destination $t_i \in V$ along a predefined s_i-t_i-path P_i. Thus, each packet consists of a triple $M_i = (s_i, t_i, P_i)$. We let $\mathcal{M} = \{M_1, M_2, M_3, ..., M_{|\mathcal{M}|}\}$ denote the set of all packets that have to be sent through the network. We assume that time is discrete and that all packets take their steps simultaneously.

A routing protocol, or *feasible packet routing schedule* for the instance $I = (G, \mathcal{M})$ is now a schedule which defines what packets enter what edges at what times (respecting transit times, bandwidths and predefined paths). The corresponding *packet routing problem* is to minimize the makespan of such a schedule, which is the last point in time when a packet reaches its destination vertex.

Dilation and Congestion. The two trivial lower bounds dilation and congestion carry easily over from unit to more general transit times and bandwidths.

Table 1. Bounds for schedules obtained in this paper depending on the minimum bandwidth b and the minimum transit time τ of all edges. The given values denote the constants in front of $(C+D)$. Note that for technical reasons (which will become clear in Section 4) the bounds do not improve when choosing τ larger than 63.

Bound on length of optimal schedule						
	$\tau = 1$	$\tau = 2$	$\tau = 5$	$\tau = 10$	...	$\tau = 63$
$b = 1$	23.40	23.21	18.54	16.37	...	4.32
$b = 2$	21.59	18.85	15.58	14.27	...	4.32
$b = 5$	16.19	14.50	12.98	12.38	...	4.32
$b = 10$	14.03	13.05	11.86	11.54	...	4.32
....	...	...	...	...	...	...
$b \to \infty$	7.63	7.63	7.63	7.63	...	4.32

For each packet M_i we define D_i to be the length of P_i and D to be the maximal length of a path, i.e., $D_i := \sum_{e \in E: e \in P_i} \tau_e$ and $D := \max_{M_i \in \mathcal{M}} D_i$. Also, for each edge $e \in E$ we define C_e to be the number of paths using e. We define the *congestion* C by $C := \max_{e \in E} \{\lceil C_e/b_e \rceil\}$. Clearly, the *dilation* D as well as the *congestion* C provide lower bounds on the length of an optimal schedule.

Remark regarding large $(C + D)$. In our analysis we will always assume that $C + D$ is large enough such that we have $\lceil k \cdot (C + D) \rceil \approx k \cdot (C + D)$ for certain constants k. This simplifies the calculations and was also implicitly used by Scheideler [24]. In order to give a fair comparison with his bounds we use this assumption as well. Moreover, we believe that also for instances where $C + D$ is small, our techniques can be used to prove good bounds for the optimal makespan. However, this would require further case distinctions which is beyond the scope of this paper.

1.2 Our Contribution

We prove bounds for the length of optimal packet routing schedules in the case of arbitrary transit times and bandwidths for the edges. For the classical setting where $b = 1$ and $\tau = 1$ we improve the best known bound of $39(C + D)$ due to Scheideler [24] to $23.4(C + D)$. Even more, for increasing b or τ our bounds improve further to up to $7.63(C+D)$ and $4.32(C+D)$, respectively. See Table 1 for an overview for some values depending on b and τ.

The key insight for our analysis is to prove and exploit a good bound for schedules with very small dilation: We show that if $D \leq \tau + 1$, then there is always a schedule of length $C + D - 1$. Note that this bound is tight, since there exist instances which require a schedule of this length (e.g., consider C packets that need to take the same path of length D). Our proof framework uses this insight in order to develop good bounds for general instances. Moreover, our approach points into a direction of promising further research: If one could prove similarly tight bounds for instances with, e.g., $D \leq k\tau + 1$ for small k, our proof framework would immediately give even better bounds for *all* instances.

This work gives the first improvement for the bounds since the result by Scheideler [24] from 1998. The paper consists of two parts: In Section 2 we prove a bound of $C + D - 1$ for instances with $D \leq \tau + 1$. Then, in Section 3, we prove our bounds for general instances, using the insights obtained in Section 2. Due to the space constraints for the full proofs we refer to our technical report [21].

1.3 Related Work

The packet routing problem and related problems are widely studied in the literature. As mentioned above, in their celebrated paper Leighton et al. show that there is always a routing schedule that finishes in $O(C + D)$ steps [12]. Leighton et al. also present an algorithm that finds such a schedule in polynomial time [13]. However, this algorithm is not suitable for practical applications since the hidden constants in the schedule length are very large (the same applies to the $O(C + D)$ existence-result in [12]).

There are various "constant factor approximation"-results in the area of packet routing which are based on the fact that a constant factor approximation on an optimal schedule for store-and-forward packet routing exists. Our improved bounds thus have implications for all of these results. We mention some examples, where store-and-forward packet routing is a subproblem that needs to be solved:

Srinivasan and Teo [25] present a constant factor approximation algorithm for packet routing with *variable paths*, i.e., in the setting where the routing paths are not part of the input, but need to be found by the algorithm. Koch et al. [11] improve and generalize this algorithm to the more general message routing problem (where each message consists of several packets). In both results, suitable paths are found that yield a constant factor approximation on the minimum of $C + D$ over all possible choices of paths. The remaining problem is then the ordinary packet routing problem. For the case that each vertex of a grid graph is the start vertex of at most one packet, Mansour and Patt-Shamir [16] prove the existence of a constant factor approximation on an optimal schedule, again by reducing the problem to an ordinary packet routing problem.

There are other result that go in the direction of the work of Leighton et al. [12]. E.g., Meyer auf der Heide et al. [9] present a randomized online-routing protocol which finishes after at most $O(C + D + \log N)$ steps if all paths are shortest paths. Rabani et al. [22] give a distributed algorithm which guarantees a bound of $O(C) + (\log^* n)^{O(\log^* n)} D + poly(\log n)$. This is improved by Ostrovsky et al. [17] who give a local protocol that needs at most $O(C + D + \log^{1+\epsilon} N)$ steps.

Packet routing is also studied in the case of special graph topologies. Leung et al. [15, chapter 37] study packet routing on certain tree topologies. Leighton, Makedon and Tollis [14] show that the permutation routing problem on an $n \times n$ grid can be solved in $2n - 2$ steps using constant size queues. Rajasekaran [23] presents several randomized algorithms for packet routing on grids.

Studying the complexity for the ordinary packet routing problem, Di Ianni shows that the delay routing problem [4] is NP-hard. The proof implies that the packet routing problem on general graphs is NP-hard as well. In [19], the

authors improve this by giving non-approximability results for the packet routing problem on several graph classes as well as algorithms for the problem on trees. The same authors present NP-hardness results and algorithms for some cases of the packet routing problem on grid graphs [20].

Busch et al. [3] study the *direct* routing problem, that is the problem of finding a routing schedule such that a packet is never delayed once it has left its start vertex. They give complexity results and algorithms for finding direct schedules. Finally, Adler et al. [1, 2] study the problem of scheduling as many packets as possible through a given network in a certain time frame. They give approximation algorithms and NP-hardness results.

With our results, we also contribute to the area of *multi-commodity flows over time* which is widely studied, e.g., see [5–8, 10]. The multi-commodity flow over time problem is equivalent to the (classical) packet routing problem with variable paths if we additionally require unit edge capacities, unit transit times, and integral flow values. Thus, packet routing with variable paths and arbitrary transit times and bandwidths corresponds to the integral multi-commodity flow problem over time.

2 Tight Bound for Instances with Small Dilation

Ideally, we would like to determine for each combination of C, D, b, and τ a tight upper bound on the maximal length of an optimal schedule for instances with these parameters. In this section, we make a first step towards this goal: We give a tight bound for instances with $D \leq \tau + 1$. As we will see in Section 3, this insight will allow us to prove good upper bounds for *all* instances.

For the sake of analysis, we replace every edge $e \in E$ with transit time τ_e by a path consisting of τ_e edges, all with the same bandwidth as e. In the resulting graph, we call every vertex that was created due to this transformation a *small* vertex. All other vertices are called *big* vertices. We say a *small path* is a path connecting two big vertices. Hence, τ is the minimum length of a small path. (Note that our assumption makes the problem slightly more general since now packets are allowed to have their start vertex "in the middle" of an edge. We introduce this generalization because it will be needed in our proof framework later.) To simplify notation later we denote by $A^{b,\tau}(C, D)$ the maximum length of an optimal schedule for an instance with minimum bandwidth b, minimum transit time τ, congestion C, and dilation D. The main result of this section is given in the following theorem.

Theorem 1. *Let I be an instance of the packet routing problem with $D \leq \tau+1$. Then there is a schedule for I whose makespan is bounded by $C+D-1$. Moreover, $A^{b,\tau}(C, D) = C + D - 1$ if $D \leq \tau + 1$.*

Proof (sketch). Let I be an instance of the packet routing problem with $D \leq \tau + 1$. Due to the assumption the path of each packet uses at most two small paths. Thus, we can divide the packets which use any small path P into two sets $\mathcal{M}_P^1$ and $\mathcal{M}_P^2$, such that $\mathcal{M}_P^1$ contains the packets for which P is the first small path, and $\mathcal{M}_P^2$ contains the packets for which P is the second small path.

In a first step, we transform I into an instance I' such that $OPT(I') \geq OPT(I)$ and for every small path P' either $\mathcal{M}^1_{P'} = \emptyset$ or $\mathcal{M}^2_{P'} = \emptyset$ (see [21] for details). While performing the necessary changes we do not change C, D or τ at all. It turns out that the resulting instance I' is an instance of the packet routing problem whose underlying graph topology is the union of directed spiders (i.e., directed trees with at most one vertex whose degree is larger than two). We then construct a schedule for each connected component which finishes after at most $C + D - 1$ steps. The idea is as follows:

Take a small path P' with bandwidth $b \geq 2$ (recall that all edges on a small path have the same bandwidth). We replace P' by b paths $P''_1, ..., P''_b$ of unit bandwidth. The packets using P' in I' are now distributed among the new paths such that no path is used by more than C packets. We do this transformation with each small path. By construction, any feasible schedule of the resulting instance can be transformed to a feasible schedule for I' with the same length.

Consider one connected component (recall: a directed spider where each edge has unit bandwidth). It was shown in [19] that for instances of the packet routing problem on directed trees one can construct a schedule of length $C + D - 1$ using path-coloring techniques. However, for this special case we can even give a simpler proof (see full version). Summarizing, since $OPT(I') \leq C + D - 1$ and $OPT(I) \leq OPT(I')$, it follows that $A^{b,\tau}(C, D) \leq C + D - 1$.

It is straight forward to construct instances I with $D \leq \tau + 1$ and $OPT(I) = C + D - 1$. For example, consider an instance with only one edge which is used by $C \cdot b$ packets. This shows that the bound is tight, i.e., $A^{b,\tau}(C, D) = C + D - 1$. □

3 High Level Ideas for General Bounds

In the previous section, we provided a tight bound for the length of optimal schedules of instances of the packet routing problem where $D \leq \tau + 1$. Unfortunately, the vast majority of instances does not fall under this category. However, in this section we provide an upper bound for *all* instances. In order to do this, we provide a framework which uses bounds for instances with small dilation (like $D \leq \tau + 1$) for proving good bounds for all instances. Using our bounds for instances with $D \leq \tau + 1$ from the previous section, we prove the following theorems.

Theorem 2. *Let I be an instance of the packet routing problem with minimum bandwidths and transit times b and τ, respectively. There is a feasible schedule for I whose length is bounded by*

$$\left(6.44 \cdot f(\tau + 1, b) + 2.11 \frac{\tau}{\tau + 1} + \delta\right)(C + D) \tag{1}$$

for a function $f(\tau + 1, b)$ which tends to 1 for increasing τ or b and with $\delta = \frac{1}{60}(\tau + 3.05 \cdot f(\tau + 1, b)(\tau + 1))$.

Note that the value δ in the theorem above increases for increasing τ. However, if $\tau \geq 63$ the following theorem gives an alternative bound.

Theorem 3. *Let I be an instance of the packet routing problem with minimum transit time $\tau \geq 63$. Then there is a feasible schedule for I whose length is bounded by $4.32(C + D)$.*

Even more, assuming that one has good upper bounds for instances with small dilation we present a framework which gives good upper bounds for *all* instances.

Theorem 4. *Let I be an instance of the packet routing problem with minimum bandwidths and transit times b and τ, respectively. For any $\ell \in \mathbb{N}$ there is a feasible schedule for I whose length is bounded by*

$$A^{b,\tau}\left(3.05\ell \cdot f(\ell, b), \ell\right) \cdot \left(\frac{2.11}{\ell} + \delta\right)(C + D) \tag{2}$$

for a function $f(\ell, b)$ which tends to 1 for increasing ℓ or b and with $\delta = \frac{1}{60}$.

For using the above theorem, it suffices to have a bound for $A^{b,\tau}\left(3.05\ell \cdot f(\ell, b), \ell\right)$ for some (small) value ℓ. As an application of this framework, the proof of Theorem 2 uses that $A^{b,\tau}(C, \tau + 1) = C + \tau$ as proven in Theorem 1 (here we choose $\ell := \tau + 1$). Also, for the important special case of unit bandwidths and unit transit time (i.e., $b = 1$ and $\tau = 1$) our framework gives the following bound.

Theorem 5. *Let I be an instance of the packet routing problem with unit transit times and unit bandwidths. Then there is a feasible schedule for S whose length is bounded by $23.4(C + D)$.*

Table 1 shows the bounds obtained by the above theorems for certain values of b and τ.

In this section we describe the high-level concepts for our proof. The complete proof requires many technical details which we give in Section 4. Our reasoning uses the concepts introduced by Scheideler [24] who proved that there is always a schedule of length $39(C + D)$ for instances with unit transit times and unit capacities. At several times in our analysis, we make use of the Lovász Local Lemma (LLL) which (in its symmetric version) states the following:

Lemma 1 (Lovász Local Lemma (LLL) [18]). *Let $E_1, \ldots, E_k$ be a series of "bad events" such that each event occurs with probability at most p and such that each event is independent of all the other events except for at most d of them. Then, if $ep(d + 1) \leq 1$, there is a nonzero probability that none of the events occurs.*

In the first part of our proof, we give a careful adaption of these concepts to the setting of arbitrary bandwidths and transit times. In the second part of the proof, we introduce our framework. Any good bound for instances with small dilation, e.g., $D \leq \tau + 1$, $D \leq 2\tau + 1$, etc., allows the framework to prove better bounds for general instances (with arbitrary dilation). We incorporate the bounds for instances with $D \leq \tau + 1$ (obtained in Section 2) into our framework. In the setting of unit bandwidths and transit times we improve the bound of $39(C + D)$ by Scheideler [24] to $23.4(C + D)$. For increasing b and/or increasing τ we can reduce the constant in front of $(C + D)$ even further.

In the sequel, we will use the concept of infeasible schedules. We call a schedule *infeasible* if in the schedule some edge e is used by more than b_e packets at a time, but the other two properties of feasible schedules are fulfilled. For ease of notation we say a schedule is infeasible if it is not necessarily feasible. We use the following strategy: We start with an infeasible schedule in which no packet is ever delayed. Denote by S_0 this schedule. We perform the following steps.

- One can enlarge S_0 by adding a random delay of at most C/b for each packet at the beginning of S_0, yielding a schedule S_1. We will show using the LLL that there are delays for the packets such that S_1 fulfills certain properties.
- Inductively, assume that we have an infeasible schedule S_i (with $i \geq 1$). Using the LLL we show that there are further refinement steps yielding a schedule S_{i+1} which is—intuitively speaking—"more feasible" than S_i.
- Eventually, we prove that there is a schedule S_k with the property that in every interval of length 64 at most $195.1b$ packets use each edge. Furthermore, we prove that the length of S_k is bounded by $1.0626(C + D)$.

Starting with the infeasible schedule S_k, we establish our framework. Let $\ell \in \mathbb{N}$. Using the LLL we show that there is an infeasible schedule S_{k+1} that can be partitioned such that in any time-interval of length ℓ at most $C^\ell_{b_e}$ packets traverse each edge e (for a constant $C^\ell_{b_e}$ to be defined later). Hence, we can turn S_{k+1} into a feasible schedule by refining each interval of length ℓ separately. In order to do this, we treat each of these intervals as a subinstance of the packet routing problem with dilation ℓ and congestion $\max_e \left\{C^\ell_{b_e}/b_e\right\}$. Hence, it suffices to have good bounds for instances with dilation $D = \ell$ in order to obtain a bound for the original instance. We use our framework with $\ell := \tau + 1$ since Theorem 1 gives a bound for instances with $D \leq \tau + 1$. Using the framework one could obtain even better bounds for general instances if one had good upper bounds for instances with slightly higher dilation, e.g., $D \leq k\tau+1$ for some small value k. In particular, the larger we can choose ℓ, the better become our bounds. This can be seen in Table 1 since for increasing τ the bounds improve. Also, if b increases the bounds improve as well. The reason is that C^ℓ_b/b decreases when b increases. Hence, the congestion in the subinstances (given as $\max_e \left\{C^\ell_{b_e}/b_e\right\}$ above) will be smaller for larger values of b.

In the following section we give a detailed technical analysis of the reasoning described above.

4 Technical Analysis

In this section we give the full proofs of the theorems and techniques stated in Section 3. First, we adapt the concepts of Scheideler [24] to the setting of arbitrary bandwidths and transit times. Then, we introduce our framework which then allows us to prove our bounds for the lengths of optimal schedules.

Let I be an instance of the packet routing problem with congestion C and dilation D. Assume that each edge has a bandwidth of at least b and each a

transit time of at least τ. Our bounds depend on these four parameters. In particular, they improve if b and τ increase. As already mentioned in Section 2, we replace every edge e with transit time τ_e by a path consisting of τ_e edges.

First, we prove the existence of the schedule S_k with the property that in every interval of length 64 at most $195.1b$ packets use each edge. We bound the length of S_k later. We define $I_0 := \max\{C, D\}$. Let $k := (\log^* I_0) - 1$[1]. We set $I_k := 4$ and $I_j := 2^{I_{j+1}}$ for all j with $1 \leq j \leq k-1$. Note that $2^{I_1} \geq I_0$. If $I_0 \leq 64$ then we define $S_k := S_0$. Hence, for the remaining reasoning for the construction of S_k we assume that $I_0 > 64$. Let S_0 be the infeasible schedule in which no packet is ever delayed. We define $D_0 := D$. We will prove the existence of schedules S_i with certain properties (with $i \geq 1$). We denote by D_i the length of S_i. Let C_i be the maximum number of packets that use an edge in each interval of length I_i^3 in the schedule S_i.

We start with the schedule S_0. We assign each packet an initial random delay. Using the LLL we prove that there are random delays such that the resulting schedule is "relatively feasible". The schedule S_1 is the schedule resulting from those "good" initial delays.

Lemma 2. *There is an infeasible schedule S_1 with the property that in every interval of length I_1^3 at most $C_1 b_e$ packets use each edge e with $C_1 := (1 + \frac{3}{I_1})I_1^3$. Also, $D_1 \leq D + C$.*

Proof (sketch). We assign each packet a random delay from the set $\{0, ..., C-1\}$. Using the LLL it can be shown that there are certain delays for the packets such that the resulting infeasible schedule fulfills the property stated in the lemma. For the full reasoning see our technical report [21]. □

Denote by S_1 the schedule whose existence was proved in Lemma 2. Given an infeasible schedule S_i we want to prove the existence of a schedule S_{i+1} which is—intuitively speaking—"more feasible" than S_i. This is again done by giving each packet random delays. We use the LLL to ensure that there are delays for the packets such that in any interval of length I_{i+1}^3 only a bounded number of packets use each edge.

Lemma 3. *Let S_i be an infeasible schedule of length D_i with the property that in every interval of length I_i^3 at most $C_i b_e$ packets use each edge e for some value $C_i \geq I_i^3$. Then there is an infeasible schedule S_{i+1} with the property that in every interval of length I_{i+1}^3 at most $C_{i+1} b_e$ packets use each edge e, with $C_{i+1} := C_i \cdot \left(1 + \frac{5.1}{I_{i+1}}\right) \cdot \frac{I_{i+1}^3}{I_i^3 - I_{i+1}^3}$. Moreover, $D_{i+1} \leq \left(1 + \frac{1}{I_i}\right) D_i$ and $C_{i+1} \geq I_{i+1}^3$.*

Proof (sketch). We split the timeline into intervals of length I_i^4. We refine the infeasible schedule S_i by enlarging each of these intervals. The schedule S_{i+1} is the concatenation of all enlarged intervals.

Consider one of these time intervals. We assign each packet an initial delay from the set of values $\{0, 1, ..., I_i^3 - I_{i+1}^3 - 1\}$. Do this with each interval. With

[1] For our purposes we define $\log^* I_0$ to be the smallest integer k such that we need to apply the log-function k times to I_0 in order to obtain a value of at most 4.

the LLL and careful bounds on the probabilities it can be shown that the resulting schedule fulfills the properties stated in the lemma. For details see our technical report [21]. □

We apply Lemma 3 iteratively until we have proven the existence of the schedule S_k with the respective properties. In particular, since $I_k = 4$ Lemma 3 shows that in S_k in every interval of length $4^3 = 64$ every edge is used by at most C_k packets. In the following lemma we bound C_k and D_k.

Lemma 4. *It holds that* $D_k < 1.0626(D + C)$ *and* $C_k < 195.1$.

The bounds can be shown with some straight forward calculations, see [21]. Note that if $I_0 \leq 64$ then by definition $S_0 = S_k$ and also $D_k = D_0 = D < 1.0626(D + C)$ and $C_k = C_0 = C < 195.1$.

4.1 Framework

Having established the existence of the schedule S_k with the above properties we introduce our framework. The idea is the following: We split the schedule S_k into intervals of length $I_k^3 = 64$. We treat each of these intervals individually as a subinstance. Let F be such an interval. At the beginning of F we assign each packet a random delay from the set $\{0, 1, ..., 63\}$. The length of the resulting schedule is hence at most 127. Let $\ell \in \mathbb{N}$. Using the LLL we show that there are random delays for the packets such that in each subinterval of length ℓ at most C_b^ℓ packets use each edge with bandwidth b (for a constant C_b^ℓ to be defined later). Denote by S_{k+1} such a schedule. Each subinterval of length ℓ can now be treated as a subinstance of the packet routing problem with dilation ℓ and congestion $\bar{C} := \max_e \left\{C_{b_e}^\ell / b_e\right\}$. Assume that we have a good bound $A^{b,\tau}\left(\bar{C}, \ell\right)$ for the maximum length of an optimal schedule for such an instance. This implies that by losing only a factor of (roughly) $A^{b,\tau}\left(\bar{C}, \ell\right)/\ell$ we can turn S_{k+1} into a feasible schedule. The length of S_{k+1} then gives us our bound on the length of an optimal schedule for the original instance. As an application of this framework we derive bounds for general instances by choosing $\ell := \tau + 1$. First, we define the values C_b^ℓ.

Definition 1. *Let* $b, \ell \in \mathbb{N}$. *Consider* $\lfloor 195.1b \rfloor$ *binary random variables* X_i *such that* $\Pr[X_i] = \frac{\ell}{64}$ *and let* $X := \sum_{i=1}^{\lfloor 195.1b \rfloor} X_i$. *We define* C_b^ℓ *to be the minimum integer such that* $\Pr\left[X > C_b^\ell\right] \cdot e \cdot \left\lceil \frac{1}{\ell} 127 \right\rceil \cdot \lfloor 195.1b \rfloor \cdot 64 \leq 1$. *We write* $\Pr\left(C_b^\ell\right) := \Pr\left[X > C_b^\ell\right]$.

Later we will split the whole time axis into intervals of length 127. We will split those again into even smaller intervals of length ℓ (or less if ℓ does not divide 127). To this end, we introduce the notion of an ℓ-partition.

Definition 2. *An* ℓ-partition *of an interval* J *with* $|J| = 127 \cdot M$ *(for an integer* M*) is a partition into* $\left\lfloor \frac{127}{\ell} \right\rfloor \cdot M$ *subintervals of length* ℓ *and* M *subintervals of length* $k \bmod \ell$. *In the sequel we call those subintervals* ℓ-subintervals.

Using the LLL, in the next lemma we show that there are random delays which turn S_k into a schedule S_{k+1} which is "almost feasible".

Lemma 5. *Let $\ell, b \in \mathbb{N}$. Assume we are given an infeasible schedule S_k of length D_k such that in every interval of length 64 each edge e is used by at most $\lfloor 195.1b \rfloor$ packets. Then there is an infeasible schedule S_{k+1} whose length is bounded by $D_{k+1} := 2D_k$ that can be ℓ-partitioned such that in every ℓ-subinterval at most $C^\ell_{b_e}$ packets use each edge e.*

Proof. Similar application of the Lovász Local Lemma as before, see [21] for details.

We can turn S_{k+1} into a feasible schedule by solving each subinstance induced by a ℓ-subinterval optimally. This increases the length of S_{k+1} at most by a factor of (roughly) $A\left(\bar{C}, \ell\right)/\ell$ with $\bar{C} = \max_e \left\{\left\lceil C^\ell_{b_e}/b_e \right\rceil\right\}$. If we have a good bound for $A\left(\bar{C}, \ell\right)$ this yields a good bound for the length of an optimal schedule for the original instance. This is shown in the following lemma.

Lemma 6. *Let I be an instance of the packet routing problem with minimum bandwidths and transit times b and τ, respectively, and let $\ell, M \in \mathbb{N}$. Assume we are given an infeasible schedule S_{k+1} for I of length $127 \cdot M$ which is ℓ-partitioned such that every ℓ-subinterval is used by at most C^ℓ_b packets. Then there is a feasible schedule for I whose length is bounded by*

$$\left\lceil \frac{127}{\ell} \right\rceil \cdot M \cdot A^{b,\tau}\left(\max_e \left\{\left\lceil C^\ell_{b_e}/b_e \right\rceil\right\}, \ell\right). \tag{3}$$

Proof. We change the given schedule S_{k+1} to a feasible schedule by refining each ℓ-subinterval of the ℓ-partition. Each ℓ-subinterval can be modeled as a subinstance of the packet routing problem. This subinstance has a dilation of at most ℓ and each edge e is used by at most $C^\ell_{b_e}$ packets. Hence, the congestion of this subinstance is $\bar{C} := \max_e \left\{\left\lceil C^\ell_{b_e}/b_e \right\rceil\right\}$. According to our assumption concerning b and τ the schedule in each ℓ-subinterval can be refined to a feasible schedule whose length is at most $A^{b,\tau}\left(\max_e \left\{\left\lceil C^\ell_{b_e}/b_e \right\rceil\right\}, \ell\right)$. This yields the bound stated in the theorem. □

Now we can prove our main theorem for the bounds derived by our framework.

Theorem 6. *Let I be an instance of the packet routing problem with minimum bandwidths and transit times b and τ, respectively. For any $\ell \in \mathbb{N}$ there is a feasible schedule for I whose length is bounded by*

$$A^{b,\tau}\left(3.05\ell \cdot f(\ell, b), \ell\right) \cdot \left(\frac{2.11}{\ell} + \delta\right)(C + D) \tag{4}$$

for a function $f(\ell, b)$ which tends to 1 for increasing ℓ or b and with $\delta \leq \frac{1}{60}$.

Proof. We introduce values $\tilde{C}^\ell_b$ which have the properties that $C^\ell_b \leq \tilde{C}^\ell_b$ and additionally $\max_e \left\{\left\lceil \tilde{C}^\ell_{b_e}/b_e \right\rceil\right\} \leq \left\lceil \tilde{C}^\ell_b/b \right\rceil$ for all ℓ and b. With Lemma 6 and some further calculations (where we use that $(C + D)$ is large as mentioned in the introduction) the claim follows. □

Theorem 1 allows us to bound the expression $A^{b,\tau}(C,\ell)$ for the case that $\ell = \tau+1$. Using this insight we can use the theorem above to derive general bounds for all packet routing instances.

Theorem 7. *Let I be an instance of the packet routing problem with minimum bandwidths and transit times b and τ, respectively. There is a feasible schedule for I whose length is bounded by*

$$\left(6.44 \cdot f(\tau+1,b) + 2.11\frac{\tau}{\tau+1} + \delta\right)(C+D) \tag{5}$$

for a function $f(\tau+1,b)$ which tends to 1 for increasing τ or b and with $\delta \leq \frac{1}{60}(\tau + 3.05 \cdot f(\tau+1,b)(\tau+1))$.

Proof. We choose $\ell := \tau+1$ in Theorem 6. Theorem 1 shows that $A^{b,\tau}(C,\tau+1) \leq C+\tau$. Calculations show that the resulting expression is upper-bounded by the expression stated in the theorem. □

Note here that – for values of $f(\tau+1,b)$ close to 1 – the formula stated in Theorem 7 gives better bounds if $\tau = 1$ than for higher values of τ. However, for small τ and b the bound of the formula improves as τ increases.

Observe that δ increases for increasing τ. This worsens the bound for very large values τ. However, later we will prove a much better bound for the case that τ is large.

Large Minimum Transit Times. Given the schedule S_k, in our framework we used the Lovász Local Lemma to prove the existence of the schedule S_{k+1}. However, we can alternatively turn t he schedule S_k to a feasible schedule directly. This is in particular useful for cases where τ is relatively large, as we will see in Corollary 1.

Theorem 8. *Let I be an instance of the packet routing problem with minimum bandwidth b and minimum transit time τ. There is feasible schedule for I whose length is at most $\frac{1}{60}(C+D) \cdot A^{b,\tau}(196,64)$.*

Proof. Recall that we proved the existence of the schedule S_k which has the property that in every interval of length 64 each edge is used by at most $C_k b = 195.1b$ packets. Hence, there is a feasible schedule for I whose length is bounded by $\frac{D_k}{64} \cdot A^{b,\tau}(\lceil C_k \rceil, 64)$. This value is at most $(D+C)/60 \cdot A^{b,\tau}(196,64)$. Note that here again we used that $(C+D)$ is large as mentioned in the introduction since then $\left\lceil \frac{D_k}{64} \right\rceil \approx \frac{D_k}{64}$. Since D_k is *strictly* smaller than $1.0626(D+C)$ there is a value N_0 such that for all C, D with $(C+D) \geq N_0$ our bounds hold. □

Using our insight gained in Theorem 1 for $A^{b,\tau}(C,\tau+1)$ allows us to prove the following corollary.

Corollary 1. *Let I be an instance of the packet routing problem with minimum bandwidth b and minimum transit time $\tau \geq 63$. Then there is always a packet routing schedule whose length is bounded by $4.32(C+D)$.*

Table 1 shows some bounds for the lengths of schedules depending on τ and b.

Unit Transit Times and Unit Bandwidths. Finally, we use the above framework to derive a bound of $23.4(C + D)$ for the case of unit transit times and unit bandwidths. This improves the bound of $39(C + D)$ proven by Scheideler [24]. First, we precisely calculate that $C_1^2 = 21$. Then we can use our framework together with Theorem 1 to derive our desired bound.

Theorem 9. *Let I be an instance of the packet routing problem with $b = 1$ and $\tau = 1$. Then there is a feasible schedule for S whose length is bounded by $23.4(C + D)$.*

Acknowledgments. We would like to thank Jan-Philipp Kappmeier, Olaf Maurer, and Jannik Matuschke for fruitful discussions and in particular Jan-Philipp Kappmeier for helpful comments on the draft.

References

1. Adler, M., Khanna, S., Rajaraman, R., Rosén, A.: Time-constrained scheduling of weighted packets on trees and meshes. Algorithmica 36, 123–152 (2003)
2. Adler, M., Sitaraman, R., Rosenberg, A., Unger, W.: Scheduling time-constrained communication in linear networks. In: Proceedings of the 10th annual ACM symposium on Parallel Algorithms and Architectures (SPAA 1998), pp. 269–278 (1998)
3. Busch, C., Magdon-Ismail, M., Mavronicolas, M., Spirakis, P.: Direct routing: Algorithms and complexity. Algorithmica 45, 45–68 (2006)
4. di Ianni, M.: Efficient delay routing. Theoretical Computer Science 196, 131–151 (1998)
5. Fleischer, L., Skutella, M.: Minimum cost flows over time without intermediate storage. In: Proceedings of the 14th Annual Symposium on Discrete Algorithms, SODA 2003 (2003)
6. Fleischer, L., Skutella, M.: Quickest flows over time. SIAM Journal on Computing 36, 1600–1630 (2007)
7. Hall, A., Hippler, S., Skutella, M.: Multicommodity flows over time: Efficient algorithms and complexity. Theoretical Computer Science 2719, 397–409 (2003)
8. Hall, A., Langkau, K., Skutella, M.: An FPTAS for quickest multicommodity flows with inflow-dependent transit times. Algorithmica 47, 299–321 (2007)
9. Meyer Auf Der Heide, F., Vocking, B.: Shortest paths routing in arbitrary networks. Journal of Algorithms 31, 105–131 (1999)
10. Hoppe, B., Tardos, É.: The quickest transshipment problem. Mathematics of Operations Research 25, 36–62 (2000)
11. Koch, R., Peis, B., Skutella, M., Wiese, A.: Real-Time Message Routing and Scheduling. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX 2009. LNCS, vol. 5687, pp. 217–230. Springer, Heidelberg (2009)
12. Leighton, F.T., Maggs, B.M., Rao, S.B.: Packet routing and job-scheduling in $O(congestion + dilation)$ steps. Combinatorica 14, 167–186 (1994)
13. Leighton, F.T., Maggs, B.M., Richa, A.W.: Fast algorithms for finding $O(congestion + dilation)$ packet routing schedules. Combinatorica 19, 375–401 (1999)
14. Leighton, F.T., Makedon, F., Tollis, I.G.: A $2n - 2$ step algorithm for routing in an $n \times n$ array with constant size queues. In: Proceedings of the 1st Annual Symposium on Parallel Algorithms and Architectures (SPAA 1989), pp. 328–335 (1989)

15. Leung, J.Y.-T.: Handbook of Scheduling: Algorithms, Models and Performance Analysis. CRC Press, Inc., Boca Raton (2004)
16. Mansour, Y., Patt-Shamir, B.: Many-to-one packet routing on grids. In: Proceedings of the 27th Annual Symposium on Theory of Computing (STOC 1995), pp. 258–267 (1995)
17. Ostrovsky, R., Rabani, Y.: Universal $O(congestion + dilation + \log^{1+\epsilon} N)$ local control packet switching algorithms. In: Proceedings of the 29th annual ACM Symposium on Theory of Computing (STOC 1997), pp. 644–653 (1997)
18. Erdős, P., Lovász, L.: Problems and results on 3-chromatic hypergraphs and some related questions. In: Infinite and Finite Sets Colloq. Math. Soc. Janos Bolyai, vol. 11, pp. 609–627. North-Holland, Amsterdam (1975)
19. Peis, B., Skutella, M., Wiese, A.: Packet routing: Complexity and algorithms. In: Bampis, E., Jansen, K. (eds.) WAOA 2009. LNCS, vol. 5893, pp. 217–228. Springer, Heidelberg (2010)
20. Peis, B., Skutella, M., Wiese, A.: Packet routing on the grid. In: López-Ortiz, A. (ed.) LATIN 2010. LNCS, vol. 6034, pp. 120–130. Springer, Heidelberg (2010)
21. Peis, B., Wiese, A.: Universal packet routing with arbitrary bandwidths and transit times. Technical Report 024-2010, Technische Universität Berlin (November 2010)
22. Rabani, Y., Tardos, É.: Distributed packet switching in arbitrary networks. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC 1996), pp. 366–375. ACM, New York (1996)
23. Rajasekaran, S.: Randomized algorithms for packet routing on the mesh. Technical Report MS-CIS-91-92, Dept. of Computer and Information Sciences, Univ. of Pennsylvania, Philadelphia, PA (1991)
24. Scheideler, C.: Universal Routing Strategies for Interconnection Networks. LNCS, vol. 1390, pp. 57–71 (1998)
25. Srinivasan, A., Teo, C.-P.: A constant-factor approximation algorithm for packet routing and balancing local vs. global criteria. SIAM Journal on Computing 30 (2001)

A Layered Graph Model and an Adaptive Layers Framework to Solve Delay-Constrained Minimum Tree Problems

Mario Ruthmair and Günther R. Raidl

Vienna University of Technology, Vienna, Austria
Institute of Computer Graphics and Algorithms
{ruthmair,raidl}@ads.tuwien.ac.at

Abstract. We present a layered graph model for delay-constrained minimum tree problems with a polynomial number of constraints which can be solved well for instances with low- to medium-sized sets of achievable delay values and not too high bounds. Layered graph models have been recently shown to frequently yield tight bounds in the context of hop- or delay-constrained network design problems. However, since the size of the layered graph heavily depends on the size of the set of achievable delay values and the corresponding delay bound the practical applicability of these models is limited. To overcome this problem we introduce an iterative strategy in which an initially small layered graph is successively extended in order to tighten lower and upper bounds until convergence to the optimal solution. Computational results show the synergetic effectiveness of both approaches outperforming existing models in nearly all cases.

1 Introduction

When designing a communication network with a central server broadcasting or multicasting information to all or some of the participants of the network, some applications, such as video conferences, require a limitation of the maximal delay from the server to each client. Beside this delay-constraint minimizing the cost of establishing the network is in most cases an important design criterion. In another example we consider a package shipping organization with a central depot guaranteeing its customers a delivery within a specified time horizon. Naturally the organization aims at minimizing the transportation costs but at the same time has to hold its promise of being in time. These network design problems can be modeled using an $\mathcal{NP}$-hard combinatorial optimization problem called *delay-constrained minimum tree (DCMT) problem* [8]. The objective is to find a minimum cost Steiner tree on a given graph with the additional constraint that the sum of delays along each path from a specified root node to any other required node must not exceed a given delay bound.

More formally, we are given an undirected graph $G = (V, E)$ with node set V, a fixed root node $s \in V$, set $R \subseteq V \setminus \{s\}$ of terminal or required nodes,

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 376–388, 2011.

set $S = V \setminus (R \cup \{s\})$ of optional Steiner nodes, edge set E, a cost function $c : E \to \mathbb{Z}^+$, a delay function $d : E \to \mathbb{Z}^+$, and a delay bound $B \in \mathbb{Z}^+$. An optimal solution to the DCMT problem is a Steiner tree $T = (V^T, E^T)$, $s \in V^T$, $R \subset V^T \subseteq V$, $E^T \subseteq E$, with minimum cost $c(T) = \sum_{e \in E^T} c_e$, satisfying the constraints $d_v = \sum_{e \in P(s,v)} d_e \leq B$, $\forall v \in R$, where $P(s, v)$ denotes the unique path from root s to node v.

There are many recent publications dedicated to the DCMT problem and its more special variants. Manyem et al. [12] showed that the problem is not in APX. Several metaheuristics have been presented, such as GRASP [17], variable neighborhood search [17,18], and path-relinking in a hybrid scatter search [18]. More heuristic approaches can be found for the variant with $R = V \setminus \{s\}$, e.g. a GRASP and a variable neighborhood descent in [15] and ant colony optimization and a variable neighborhood search in [16]. Furthermore, preprocessing methods are presented in [16] reducing the size of the input graph significantly.

Exact methods for the DCMT problem based on integer linear programming (ILP) have been explored by Leggieri et al. [9] who describe a compact extended node-based formulation using lifted Miller-Tucker-Zemlin inequalities. Since these Big-M formulations usually yield rather weak linear programming (LP) bounds they improve it by adding directed connection cuts. In [3] Gouveia et al. transform a DCMT problem variant called *Hop-Constrained Minimum Spanning Tree (HCMST) Problem* where $d_e = 1$, $\forall e \in E$, and $R = V \setminus \{s\}$, to an equivalent Steiner tree problem (STP) [2] on an appropriate layered graph without additional constraints. The intensively studied STP can then be solved by any existing approach for directed graphs. In [3] a classic directed connection cut formulation on this layered graph is solved by an efficient branch-and-cut algorithm. This formulation has been shown to be stronger than the HCMST formulation in [4] only modeling the constrained shortest path subproblems on a layered graph. ILP approaches for the DCMT problem with $R = V \setminus \{s\}$ have been examined by Gouveia et al. in [6] based on the concept of constrained shortest paths utilized in column generation and Lagrangian relaxation methods. Similarly to [4] a third approach reformulates the constrained shortest path subproblems on a layered graph and solves them using a multi commodity flow (MCF) formulation. Since the size of the layered graph and therefore the efficiency of the according model heavily depends on the number of achievable delay values (see Section 2) this approach can in practice only be used for instances with a reasonably small set of delay values and rather low bounds. Additionally, MCF models usually suffer from the huge amount of flow variables used in the ILP formulation altogether leading to a slow and memory-intensive solving process. Nevertheless solving these layered graph models turned out to be very effective on certain classes of instances, not only for DCMT problems, but e.g. for the hop-constrained connected facility location problem as well, see [10].

The success of the layered graph transformation for some special variants of the DCMT problem leads us to a further investigation of this approach. First, we introduce an efficient ILP model utilizing the special structure of the layered graph and improving the computational performance compared to existing

models. However, the problems with huge sets of achievable delay values and high bounds still exist although much harder instances can now be tackled. To substantially improve this situation we present a new iterative strategy based on solving the problem on smaller layered graphs yielding lower and upper bounds to the optimal costs. By extending these simplified graphs appropriately the bounds are tightened to finally converge to an optimal solution. Compared to our first approach the iterative framework consumes substantially less memory. More generally, our strategy can in principle also be applied to other problems with delay or weight constraints that can be modeled on layered graphs.

The rest of the article is organized as follows: Section 2 describes the transformation to the layered digraph, Section 3 presents the ILP model on this graph and Section 4 shows some theoretical results utilized in the adaptive layers framework in Section 5. Section 6 discusses computational results and Section 7 concludes the article and sketches future work.

2 Transformation to the Steiner Arborescence Problem on Layered Digraphs

Similarly to [6,3] we transform the original graph $G = (V, E)$ to a layered digraph $G_L = (V_L, A_L)$. The node set $V_L = \{s\} \cup S_L \cup R_L$ includes Steiner nodes $S_L = \{i_l : i \in R \cup S,\ 1 \leq l \leq (B-1)\}$ and required nodes $R_L = \{i_B : i \in R\}$. The arc set $A_L = A_s \cup A_g \cup A_z$ consists of root arcs $A_s = \{(s, i_{d_{si}}) : \{s, i\} \in E\}$, general arcs $A_g = \{(i_l, j_{l+d_{ij}}), (j_l, i_{l+d_{ij}}) : \{i, j\} \in E,\ i, j \neq s,\ 1 \leq l \leq (B - d_{ij})\}$ and zero arcs $A_z = \{(i_l, i_B) : i \in R,\ 1 \leq l \leq (B-1)\}$. Arc delays d_{ij} are not needed in G_L since they are implicitly contained in the layered structure: node i_l in G_L represents node i in G with $d_i = l$. The arc costs in A_s and A_g equal the costs of corresponding edges in E, arcs A_z get assigned zero costs. Fig. 1(a) and 1(b) demonstrate the transformation. Usually, G_L can be reduced by the following preprocessing rule: if a Steiner node $v \in S_L$ has no incoming or no outgoing arcs it is removed together with all incident arcs. This preprocessing is able to reduce the number of Steiner nodes and arcs significantly especially for instances with a broad range of delay values, see Fig. 1(c) and Table 1. Further preprocessing methods for Steiner trees can be found in [7,11].

The *Steiner Arborescence Problem* on G_L is to find a Steiner tree $T_L = (V_L^T, A_L^T)$ rooted in $s \in V_L^T$ with $R_L \subset V_L^T \subseteq V_L$, $A_L^T \subseteq A_L$ and minimal arc costs $c_L^T = \sum_{a \in A_L^T} c_a$. An optimal Steiner arborescence $T_{L,\text{opt}}$ on G_L corresponds to an optimal Steiner tree T_{opt} on G, moreover $c(T_{L,\text{opt}}) = c(T_{\text{opt}})$. This has been shown in [3] for the HCMST problem and can be generalized to the DCMT problem in a natural way. We simply transform $T_{L,\text{opt}}$ to T_{opt} by removing all zero arcs $(i_l, i_B) \in A_L^T$ together with their target nodes and rename all nodes $i_l \in V_L^T$ to i. Fig. 1(c) and 1(d) show the optimal solution to the DCMT problem on the example graph in Fig. 1(a).

There are many existing approaches for efficiently solving the Steiner tree problem on graphs, e.g. [7,14,1]. All general ILP tree models either need additional variables or an exponential number of constraints. In our case we are

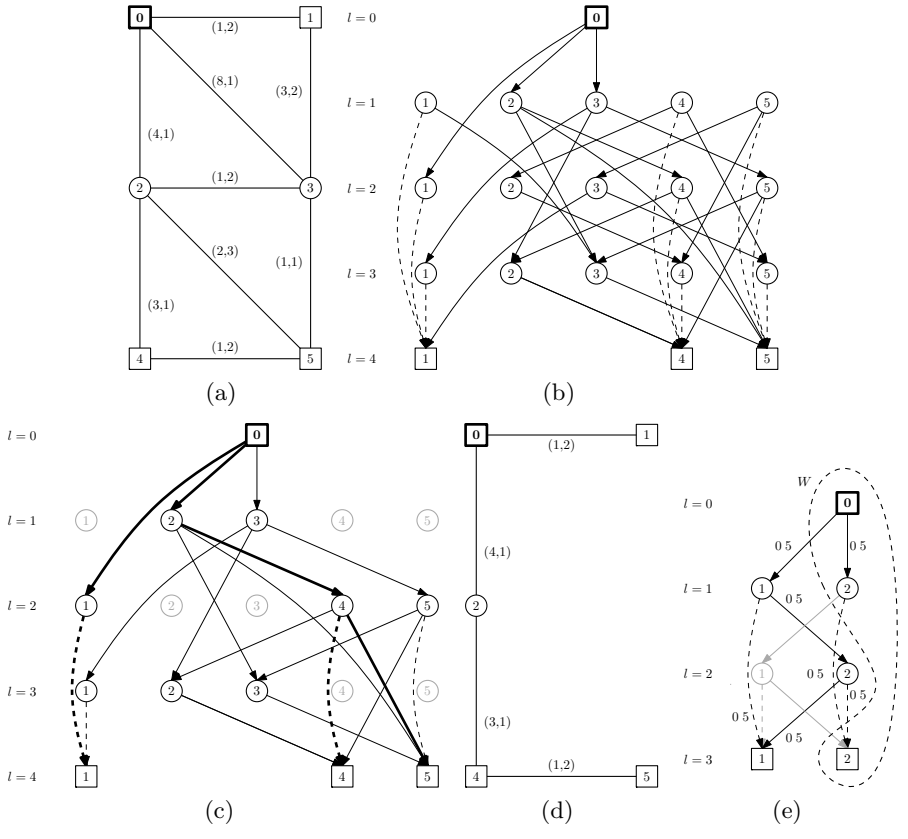


Fig. 1. Example graph in (a) with edge labels (*cost*, *delay*) and root node 0. Squared nodes denote terminal nodes. Corresponding layered digraph in (b) for $B = 4$ (arc costs are omitted). Preprocessed graph G_L in (c) with optimal solution denoted by bold arcs. Optimal tree T in G in (d) with $c(T) = 9$. Example of an LP solution in (e) where a directed connection cut inequality based on set W tightens the LP relaxation (arc labels denote the corresponding y-values, grayed out arcs mean $y = 0$).

lucky to work on a special graph structure: the layered digraph is acyclic. This property makes it possible to model the problem effectively with a polynomial number of constraints without additional variables, see [13,15] and Section 3. Zelikovsky et al. [19] present approximation algorithms for the Steiner tree problem in acyclic digraphs.

3 ILP Model on the Layered Digraph

We use binary variables x_e as design variables for edges $e \in E$ indicating whether the edge is included in the solution T ($x_e = 1$) or not ($x_e = 0$). Similarly, we

use non-negative variables $y_{i_l j_k}$ for arcs $(i_l, j_k) \in A_L$ in T_L. Adapting the hop-indexed model for the HCMST problem in [5] to the DCMT problem leads to the Steiner Arborescence Layered (SAL) model:

$$\min \quad \sum_{e \in E} c_e x_e \tag{1}$$

$$\text{s.t.} \quad \sum_{(i_l, j_B) \in A_L} y_{i_l j_B} = 1 \quad \forall j \in R \tag{2}$$

$$\sum_{(k_{l-d_{ki}}, i_l) \in A_L, k \neq j} y_{k_{l-d_{ki}} i_l} \geq y_{i_l j_{l+d_{ij}}} \quad \forall (i_l, j_{l+d_{ij}}) \in A_g \tag{3}$$

$$\sum_{(k_{l-d_{ki}}, i_l) \in A_L} y_{k_{l-d_{ki}} i_l} = y_{i_l i_B} \quad \forall (i_l, i_B) \in A_z \tag{4}$$

$$y_{s i_{d_{si}}} = x_e \quad \forall e = \{s, i\} \in E \tag{5}$$

$$\sum_{(i_l, j_{l+d_{ij}}) \in A_L} y_{i_l j_{l+d_{ij}}} + \sum_{(j_l, i_{l+d_{ij}}) \in A_L} y_{j_l i_{l+d_{ij}}} = x_e \quad \forall e = \{i, j\} \in E,\ i, j \neq s \tag{6}$$

$$y_{i_l j_k} \geq 0 \quad \forall (i_l, j_l) \in A_L \tag{7}$$

$$x_e \in \{0, 1\} \quad \forall e \in E \tag{8}$$

Constraints (2) ensure that each terminal node in G_L has exactly one incoming arc. The connectivity constraints (3) make sure that if there is an arc going out of a Steiner node there has to be an incoming arc, too. Constraints (4) force the use of the zero arc if there is an incoming arc. Together with equalities (2) this leads to the use of at most one Steiner node of $\{i_l : 1 \leq l \leq (B-1)\},\ \forall i \in R$. Equalities (5) and (6) link the directed arc variables y in G_L to the corresponding undirected edge variables x in G. By relaxing the integrality constraints (8) we obtain the corresponding linear program SAL_{LP}.

Theorem 1. *Model SAL can be used to solve the DCMT problem.*

Proof. Constraints (2) force exactly one incoming arc for each terminal node $j \in R_L$ in layer B of G_L. We have to show that all terminal nodes are connected to the root node s. If the incoming arc (i, j) originates in s we are done. Otherwise constraints (3) and (4) ensure an incoming arc to i. Due to the acyclicity and the layered structure of G_L the source of an arc can only be in a lower layer than the target. Repeating this argumentation for node i extends the path in a backtracking way to the root node in layer 0. The union of all such paths forms a connected acyclic graph including all terminal nodes. □

We optionally add directed connection cut inequalities

$$\sum_{(i_l, j_k) \in A_L,\ i_l \in W,\ j_k \notin W} y_{i_l j_k} \geq 1 \qquad \forall W \subset V_L,\ s \in W,\ (V_L \setminus W) \cap R_L \neq \emptyset \tag{9}$$

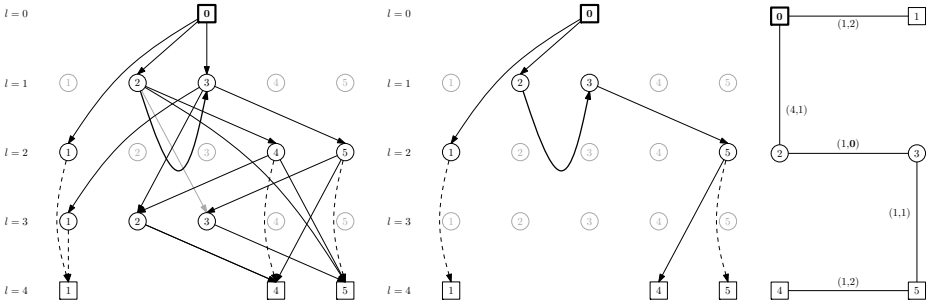


Fig. 2. A layered graph G'_L is shown on the left derived from G_L by redirecting arc $(2_1, 3_3)$ to node 3 on layer 1. In the middle the optimal solution $T'_{L,\text{opt}}$ in G'_L and on the right the reverse transformed infeasible solution T'_{opt} in G with $c(T'_{\text{opt}}) = 8$ is shown. The delay of edge $(2, 3)$ in G is decreased to 0.

to model *SAL* to further tighten the LP relaxation, see Fig. 1(e) for an example, and denote this extended model SAL^{dcut}. The optimal LP value of this model is at least as high as the one of the MCF model in [6] and there are cases in which the LP value is strictly better. This result has been shown by Gouveia et al. [3] for the HCMST problem, and the proof can be trivially adapted for our DCMT case.

The number of variables and constraints of model *SAL* can be estimated by Equations (10) and (11) showing the high dependency on the delay bound B. Therefore B is crucial for the performance and memory consumption of this model which can be clearly observed in the experimental results, see Section 6.

$$|variables| = |E| + |A_L| = \mathcal{O}(|E| \cdot B) \tag{10}$$

$$|constraints| = |R| + |A_g| + |A_z| + 2|E| + |A_L| = \mathcal{O}(|V| + |E| \cdot B) \tag{11}$$

To partly overcome this drawback we introduce the *Adaptive Layers Framework (ALF)* in Section 5 based on the theoretical results presented in the following. In principle *ALF* calculates lower and upper bounds to the optimal costs in reduced layered graphs and iteratively closes the gap by extending these graphs appropriately until the two bounds are equal.

4 Lower and Upper Bounds by Redirecting Arcs

We define the length of the shortest delay path to each node in the original graph G:

$$d_v^{\min} := \min_{P(s,v)} \sum_{e \in P(s,v)} d_e, \quad \forall v \in V \setminus \{s\}$$

In G_L we now consider an arc $(u_{l-d_{uv}}, v_l) \in A_s \cup A_g$, $u_{l-d_{uv}} \in V_L$, $v_l \in S_L$, $d_v^{\min} < l < B$ and redirect its target to a node $v_k \in S_L$ on a lower layer $k < l$. Since $l > d_v^{\min}$ there always exists a feasible node v_k with $k < l$. We denote the resulting graph G'_L.

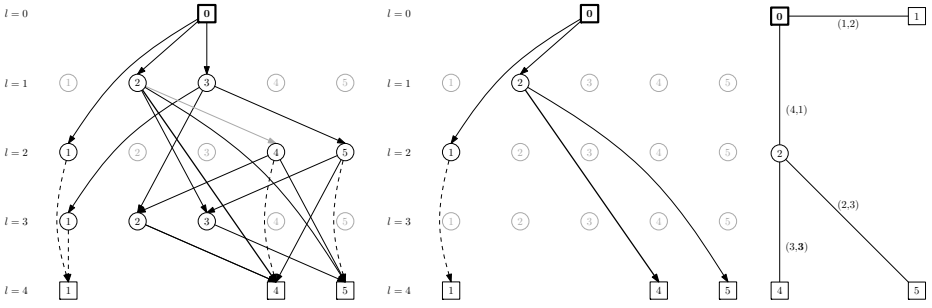


Fig. 3. A layered graph G''_L is shown on the left derived from G_L by redirecting arc $(2_1, 4_2)$ to node 4 on layer 4. In the middle the optimal solution $T''_{L,\text{opt}}$ in G''_L and on the right the reverse transformed solution T''_{opt} in G with $c(T''_{\text{opt}}) = 10$ is shown. The delay of edge $(2, 4)$ in G is increased to 3.

Lemma 1. *Let $T_{L,\text{opt}}$ and $T'_{L,\text{opt}}$ be optimal solutions to the Steiner arborescence problem on G_L and G'_L, respectively. Furthermore we denote by T_{opt} and T'_{opt} the corresponding reverse transformed trees in G, respectively. Then $c(T'_{\text{opt}}) \leq c(T_{\text{opt}})$.*

Proof. Redirecting an arc in G_L from target v_l to node v_k on a lower layer $k < l$ corresponds to a decrease of the related edge delay in G. Therefore, the solution space may be extended since it may now be easier to satisfy the delay bound. The optimal solution T_{opt} still stays feasible but one or more of the new solutions may have less cost than T_{opt}, so $c(T'_{\text{opt}}) \leq c(T_{\text{opt}})$. □

Fig. 2 shows layered graph G'_L derived from the previous example graph in Fig. 1(c). One arc is redirected to a new target on a lower layer. The optimal solution $T'_{L,\text{opt}}$ in G'_L has less cost than the optimal solution $T_{L,\text{opt}}$ in G_L. Therefore T'_{opt} cannot be feasible for the DCMT problem on G otherwise T_{opt} would not be optimal. On the other hand if T'_{opt} would be feasible Lemma 2 applies.

Lemma 2. *If T'_{opt} is feasible for the DCMT problem on G then it is optimal.*

Proof. According to Lemma 1, $c(T'_{\text{opt}}) \leq c(T_{\text{opt}})$. If T'_{opt} is feasible for the DCMT problem on G then $c(T'_{\text{opt}}) = c(T_{\text{opt}})$. Therefore T'_{opt} is an optimal solution to the original problem. □

If we redirect an arc to a target node on a higher layer instead of a lower one and denote the resulting graph G''_L, Lemma 3 holds.

Lemma 3. *Let T''_L be any feasible solution in G''_L and T'' the corresponding reverse transformed tree in G. Then T'' is feasible for the DCMT problem on G. Furthermore, $c(T''_{\text{opt}}) \geq c(T_{\text{opt}})$.*

Proof. Redirecting an arc in G_L to a new target node on a higher layer corresponds to increasing the respective edge delay in G. Therefore it may be harder

to satisfy the delay bound and the solution space may be pruned. Nevertheless all feasible solutions T'' stay feasible for the original DCMT problem on G since replacing the modified edge delay by their original smaller one cannot violate the delay bound. However, former optimal solutions may now be infeasible in G''_L, so $c(T''_{\text{opt}}) \geq c(T_{\text{opt}})$. □

Figure 3 shows an example to this case. By redirecting arc $(2_1, 4_2)$ to a higher layer the former optimal solution $T_{L,\text{opt}}$ now is not valid anymore in G''_L. Here the new optimal solution T''_{opt} in G has higher cost and therefore provides an upper bound to T_{opt}.

5 Adaptive Layers Framework (ALF)

To reduce the size of a layered graph G_L we consider a node $v_l \in S_L,\ d_v^{\min} < l < B$ and redirect all incoming arcs $(u_{l-d_{uv}}, v_l) \in A_L,\ u_{l-d_{uv}} \in V_L$, to a node v_k on a different layer $k \neq l$. Then we can safely remove v_l together with all outgoing arcs from G_L since it cannot be reached from the root s anymore and therefore cannot be part of a solution. If we want to obtain a lower bound the layer k of the new target node v_k is set to $\max_{d_v^{\min} \leq i < l}\{i : v_i \in V_L\}$, for an upper bound $k = \min_{l < i \leq B}\{i : v_i \in V_L\}$. In other words, we want to minimize the difference between the original and the redirected target layer. Repeating this redirection process for further Steiner nodes results in a sequence of layered graphs with monotonically decreasing size. In contrast to this reduction process *ALF* goes the other way and starts with the smallest possible layered graph providing a feasible solution to the model *SAL* and extends it iteratively to tighten the bounds. Node set V_L^0 of the initial layered graph G_L^0 just contains root node s, terminal nodes R_L, Steiner nodes $v_{d_v^{\min}},\ v \in R \cup S$, on the lowest feasible layer and Steiner nodes $v_{B-1},\ v \in S$. If necessary, arcs are redirected to the next available layers depending on the desired bound.

The next step is to compute optimal LP and integer solutions $T^i_{L,\text{LP}}$ and $T^i_{L,\text{opt}}$ of model $SAL_{(\text{LP})}$ on the current layered graph G^i_L. For all redirected arcs $(u_l, v_k) \in A^i_L$ with $y_{u_l v_k} > 0$ in $T^i_{L,\text{LP}}$ or $T^i_{L,\text{opt}}$ we extend G^i_L by adding the node $v_{l+d_{uv}}$ together with related outgoing arcs. If necessary, existing arcs pointing to a node $v_l,\ 1 \leq l \leq B$, are modified to either prevent a former redirection or to reduce the difference between the original and the current target layer. The resulting graph is denoted G^{i+1}_L. Applying Lemma 1 and 3 we know that $c(T^{i+1}_{L,\text{opt}}) \geq c(T^i_{L,\text{opt}})$ if redirecting to lower layers and $c(T^{i+1}_{L,\text{opt}}) \leq c(T^i_{L,\text{opt}})$ otherwise. These steps are now repeated on G^{i+1}_L and further graphs until the two bounds match. Algorithm 1 shows this iterative solving process.

When redirecting to lower layers we have to consider that the resulting graph G^i_L does not necessarily need to be acyclic anymore. Therefore $T^i_{L,\text{LP}}$ or $T^i_{L,\text{opt}}$ of model *SAL* may be unconnected and contain cycles. Adding violated directed connection cut inequalities prevents these cycles and therefore may lift the lower bound. When redirecting the arcs to higher layers we are not faced with this

Algorithm 1. Adaptive Layers Framework (ALF)

Input: graph $G = (V, E)$
Output: an optimal solution T_{opt} to the DCMT problem
$V_L^0 = s \cup \{v_{d_v^{\min}} : v \in R \cup S\} \cup \{v_{B-1} : v \in S\} \cup R_L$ `// initial node set`
$LB = 0,\ UB = \infty$ `// lower and upper bounds`
redirect = down `// arc redirection` $\in \{\text{down}, \text{up}\}$
$i = 0$
while $LB \neq UB$ **do**
  build G_L^i depending on V_L^i and *redirect*
  $T_{L,\text{LP}}^i = solve(SAL_{\text{LP}}^{(dcut)}(G_L^i))$ `// solve LP`
  $T_{L,\text{opt}}^i = solve(SAL^{(dcut)}(G_L^i))$ `// solve ILP`
  transform $T_{L,\text{opt}}^i$ to T_{opt}^i on G
  if (T_{opt}^i *is feasible*) $\wedge$ (*redirect* == up) **then** $T_{\text{opt}}^i = improve(T_{\text{opt}}^i)$
  if (T_{opt}^i *is feasible*) $\wedge\, (c(T_{\text{opt}}^i) < UB)$ **then** $UB = c(T_{\text{opt}}^i),\ T_{\text{opt}} = T_{\text{opt}}^i$
  if *redirect* == down **then** $LB = c(T_{\text{opt}}^i)$
  $V_L^{i+1} = V_L^i$ `// extend layered graph` G_L^i
  forall $(u_l, v_k) \in T_{L,\text{LP}}^i \cup T_{L,\text{opt}}^i$ **do**
    if $(y_{u_l v_k} > 0) \wedge (k - l \neq d_{uv})$ **then** $V_L^{i+1} = V_L^{i+1} \cup \{v_{l+d_{uv}}\}$
  switch *redirect*, $i = i + 1$
return T_{opt}

problem, so connection cuts cannot improve generated upper bounds in this case. Additionally, every time we obtain a solution feasible in G we try to improve it by heuristic methods to further tighten the global upper bound.

Theorem 2. *Algorithm 1 terminates.*

Proof. As long as the optimal solution $T_{L,\text{opt}}^i$ in graph G_L^i is infeasible for the DCMT problem when calculating a lower bound, $T_{L,\text{opt}}^i$ must contain a redirected arc. Adding an appropriate node v_l to G_L^{i+1} prevents the redirection leading to a new solution $T_{L,\text{opt}}^{i+1}$. In worst case all nodes $v_l \in S_L$ are added to the layered graph resulting in the original graph G_L with optimal solution $T_{L,\text{opt}}$. Since $|S_L|$ is finite the number of iterations is bounded. □

More generally, ALF can be (easily) adapted to work for other problems with delay or weight constraints that can be modeled on a directed layered graph. One just has to replace model $SAL^{(dcut)}$ by an appropriate model for the considered problem on the layered graph. To enhance upper bounds some problem specific heuristics could be provided, too.

6 Computational Results

We compared our model $SAL^{(dcut)}$ and framework $ALF^{(dcut)}$ to three existing approaches: the Miller-Tucker-Zemlin based model (MTZ), its variant with additional connection cuts from [9] (MTZ^{dcut}) and the MCF formulation on layered

Table 1. Average sizes of preprocessed graphs G, full layered graphs G_L and final layered graphs G_L^I in $ALF^{(dcut)}$ after I iterations (B: delay bound)

						$\overline{I}$		$\overline{V_L^I}$		$\overline{A_L^I}$	
Set	B	$\overline{\lvert V\rvert}$	$\overline{\lvert E\rvert}$	$\overline{\lvert V_L\rvert}$	$\overline{\lvert A_L\rvert}$	ALF	ALF^{dcut}	ALF	ALF^{dcut}	ALF	ALF^{dcut}
R1000	1000	41	401	34219	433632	15	13	681	642	7565	7167
	1500	41	414	54219	843142	16	14	895	852	12363	11909
	2000	41	414	74219	1256542	15	14	760	687	12274	11019
	2500	41	414	94219	1669942	12	11	617	565	10634	9693
C1000	1000	41	572	34221	537767	16	13	831	725	11683	10256
	1500	41	589	54221	1106116	15	13	1384	1170	25110	21519
	2000	41	589	74221	1679516	17	17	1687	1731	34899	35896
	2500	41	589	94221	2252916	13	12	1762	1510	40291	34531
E1000	1000	41	632	34220	565509	18	17	1298	1171	19472	17379
	1500	41	668	54220	1215032	15	13	1685	1453	33291	28787
	2000	41	668	74220	1874432	11	11	1785	1890	42255	44663
	2500	41	668	94220	2533832	10	11	1829	1932	49036	51358

graphs from [6] (*MCFL*). Tests were performed on complete spanning tree instances with 41 nodes introduced in [6]. The three main instance sets R, C and E each have different graph structures defined by their edge cost functions: R has random edge costs, C and E both have Euclidean costs fixing the source s near the center and near the border, respectively. Each main instance set consists of different subsets of five input graphs varying in the number of possible discrete edge delay values, e.g. C5 denotes the set of instances with five different integer delay values $d_e \in \{1, ..., 5\}$, $\forall e \in E$. All tests have been executed on a single core of a multicore system consisting of Intel Xeon E5540 processors with 2.53 GHz and about 3 GB RAM per core. We used IBM ILOG CPLEX 12.1 to solve the (I)LP models. The y variables in model $SAL^{(dcut)}$ and the flow variables in model *MCFL* are declared Boolean since the CPLEX presolver benefits from integrality of these variables and therefore can significantly reduce the model. To reduce the size of the input graphs all preprocessing methods presented in [16] are applied. Solutions obtained in ALF which are feasible in G are improved by a variable neighborhood descent introduced in [15] to further tighten upper bounds. Table 1 exemplarily shows graph sizes of the instance sets with the largest sets of achievable delay values. We list average sizes of preprocessed input graphs, full layered graphs used in model $SAL^{(dcut)}$, and final layered graphs in iteration I of $ALF^{(dcut)}$ when either an optimal solution is found or the time limit is reached.

When iteratively solving (I)LP models in the ALF framework we can provide tight upper bounds to CPLEX obtained in previous calculations supporting the presolving phase and pruning of the branch-and-bound tree. According to Lemma 2 it would be enough to iteratively compute lower bounds in ALF. But repeated switching between lower and upper bound turned out to work well in practice speeding up the convergence. Finally, in case of limited runtime ALF usually yields small gaps and obtains feasible solutions by computing both bounds. Some tests were performed to initialize the first layered graph in a more sophisticated way, e.g. based on heuristic solutions, but following the proposed trivial way mostly yields the best results.

Table 2. Comparison of different ILP models on test sets from [6] (B: delay bound, O: number of optimal solutions (out of 5), *gap*: average gap in percent, *time*: median CPU time in seconds; time limit: 10000 seconds; best results are printed bold)

		MTZ			*MTZ*^{dcut}			*MCFL*			*SAL*			*SAL*^{dcut}			*ALF*			*ALF*^{dcut}		
Set	B	O	*gap*	*time*	O	*gap*	*time*	O	*gap*	*time*	O	*gap*	*time*	O	*gap*	*time*	O	*gap*	*time*	O	*gap*	*time*
R2	3	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**
	5	**5**	**0.0**	11	**5**	**0.0**	13	**5**	**0.0**	19	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1
	7	**5**	**0.0**	7	**5**	**0.0**	6	**5**	**0.0**	91	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	1
	9	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	677	**5**	**0.0**	1	**5**	**0.0**	1	**5**	**0.0**	2	**5**	**0.0**	2
C2	3	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**
	5	3	1.1	6403	3	1.1	1627	**5**	**0.0**	9	**5**	**0.0**	2	**5**	**0.0**	**0**	**5**	**0.0**	5	**5**	**0.0**	3
	7	4	0.9	3487	**5**	**0.0**	335	**5**	**0.0**	891	**5**	**0.0**	5	**5**	**0.0**	**2**	**5**	**0.0**	10	**5**	**0.0**	6
	9	**5**	**0.0**	1074	**5**	**0.0**	31	**5**	**0.0**	1794	**5**	**0.0**	7	**5**	**0.0**	**3**	**5**	**0.0**	20	**5**	**0.0**	16
E2	3	**5**	**0.0**	10	**5**	**0.0**	11	**5**	**0.0**	1	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	1
	5	0	9.9	10000	0	11.6	10000	**5**	**0.0**	274	**5**	**0.0**	19	**5**	**0.0**	**4**	**5**	**0.0**	29	**5**	**0.0**	34
	7	0	10.6	10000	0	9.7	10000	0	82.1	10000	**5**	**0.0**	375	**5**	**0.0**	**19**	**5**	**0.0**	1512	**5**	**0.0**	715
	9	0	7.4	10000	0	5.4	10000	0	100.0	10000	**5**	**0.0**	1155	**5**	**0.0**	**44**	4	0.3	1944	**5**	**0.0**	843
R5	6	**5**	**0.0**	15	**5**	**0.0**	17	**5**	**0.0**	17	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**
	8	**5**	**0.0**	44	**5**	**0.0**	28	**5**	**0.0**	195	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	1
	10	**5**	**0.0**	49	4	1.1	30	**5**	**0.0**	561	**5**	**0.0**	**1**	**5**	**0.0**	**1**	**5**	**0.0**	2	**5**	**0.0**	2
	12	**5**	**0.0**	24	**5**	**0.0**	16	**5**	**0.0**	1119	**5**	**0.0**	**1**	**5**	**0.0**	**1**	**5**	**0.0**	2	**5**	**0.0**	2
C5	6	**5**	**0.0**	24	**5**	**0.0**	34	**5**	**0.0**	8	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	1
	8	4	1.1	714	4	0.7	564	**5**	**0.0**	109	**5**	**0.0**	1	**5**	**0.0**	**0**	**5**	**0.0**	2	**5**	**0.0**	1
	10	2	2.4	10000	2	1.5	10000	3	40.0	3269	**5**	**0.0**	4	**5**	**0.0**	**2**	**5**	**0.0**	15	**5**	**0.0**	10
	12	1	2.2	10000	**5**	**0.0**	1416	4	0.2	6723	**5**	**0.0**	9	**5**	**0.0**	**6**	**5**	**0.0**	36	**5**	**0.0**	32
E5	6	**5**	**0.0**	2437	**5**	**0.0**	7258	**5**	**0.0**	77	**5**	**0.0**	2	**5**	**0.0**	**1**	**5**	**0.0**	7	**5**	**0.0**	7
	8	0	7.5	10000	0	7.0	10000	**5**	**0.0**	897	**5**	**0.0**	6	**5**	**0.0**	**3**	**5**	**0.0**	29	**5**	**0.0**	14
	10	0	9.1	10000	0	8.5	10000	3	40.0	4067	**5**	**0.0**	18	**5**	**0.0**	**13**	**5**	**0.0**	62	**5**	**0.0**	54
	12	0	9.3	10000	0	8.0	10000	1	80.0	10000	**5**	**0.0**	125	**5**	**0.0**	**52**	**5**	**0.0**	686	**5**	**0.0**	335
R10	10	**5**	**0.0**	45	**5**	**0.0**	77	**5**	**0.0**	149	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	1
	15	**5**	**0.0**	129	**5**	**0.0**	322	4	20.0	2291	**5**	**0.0**	**1**	**5**	**0.0**	**1**	**5**	**0.0**	3	**5**	**0.0**	3
	20	**5**	**0.0**	63	**5**	**0.0**	90	3	40.0	8539	**5**	**0.0**	**1**	**5**	**0.0**	**1**	**5**	**0.0**	5	**5**	**0.0**	4
	25	**5**	**0.0**	28	4	0.6	56	0	100.0	10000	**5**	**0.0**	**3**	**5**	**0.0**	**3**	**5**	**0.0**	8	**5**	**0.0**	5
C10	10	**5**	**0.0**	70	**5**	**0.0**	45	**5**	**0.0**	220	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	2	**5**	**0.0**	2
	15	4	0.8	1034	**5**	**0.0**	481	3	40.0	3212	**5**	**0.0**	**2**	**5**	**0.0**	**2**	**5**	**0.0**	7	**5**	**0.0**	5
	20	2	2.6	10000	3	0.8	3485	0	100.0	10000	**5**	**0.0**	25	**5**	**0.0**	**18**	**5**	**0.0**	37	**5**	**0.0**	32
	25	1	2.2	10000	**5**	**0.0**	2801	0	100.0	10000	**5**	**0.0**	141	**5**	**0.0**	**84**	**5**	**0.0**	200	**5**	**0.0**	167
E10	10	4	0.5	2590	**5**	**0.0**	2918	**5**	**0.0**	301	**5**	**0.0**	**1**	**5**	**0.0**	**1**	**5**	**0.0**	5	**5**	**0.0**	6
	15	0	10.0	10000	0	11.0	10000	0	68.0	10000	**5**	**0.0**	**14**	**5**	**0.0**	24	**5**	**0.0**	140	**5**	**0.0**	112
	20	0	10.1	10000	0	10.0	10000	0	100.0	10000	**5**	**0.0**	590	**5**	**0.0**	**267**	**5**	**0.0**	3320	**5**	**0.0**	2089
	25	0	9.1	10000	0	7.1	10000	0	100.0	10000	2	1.5	10000	**5**	**0.0**	**1105**	2	1.6	10000	3	1.9	9043
R100	100	4	4.5	129	4	4.8	177	0	100.0	10000	**5**	**0.0**	10	**5**	**0.0**	**8**	**5**	**0.0**	14	**5**	**0.0**	11
	150	4	5.6	56	4	6.5	131	0	100.0	10000	**5**	**0.0**	14	**5**	**0.0**	15	**5**	**0.0**	**5**	**5**	**0.0**	6
	200	4	1.3	34	4	2.6	143	0	100.0	10000	**5**	**0.0**	27	**5**	**0.0**	34	**5**	**0.0**	**10**	**5**	**0.0**	11
	250	**5**	**0.0**	9	**5**	**0.0**	9	0	100.0	10000	**5**	**0.0**	70	**5**	**0.0**	66	**5**	**0.0**	12	**5**	**0.0**	**6**
C100	100	2	3.0	10000	3	2.2	8651	0	100.0	10000	**5**	**0.0**	**67**	**5**	**0.0**	122	**5**	**0.0**	89	**5**	**0.0**	87
	150	0	4.1	10000	2	1.2	10000	0	100.0	10000	**5**	**0.0**	1027	4	11.1	1824	**5**	**0.0**	**294**	**5**	**0.0**	389
	200	1	2.4	10000	3	0.7	2539	0	100.0	10000	2	2.3	10000	1	54.0	10000	**5**	**0.0**	1785	**5**	**0.0**	**1756**
	250	3	1.3	3423	**5**	**0.0**	**141**	0	100.0	10000	1	5.0	10000	0	59.3	10000	4	0.2	2007	4	0.1	4139
E100	100	0	8.3	10000	0	7.3	10000	0	100.0	10000	**5**	**0.0**	886	**5**	**0.0**	1211	**5**	**0.0**	**717**	**5**	**0.0**	1150
	150	0	12.1	10000	0	9.4	10000	0	100.0	10000	1	6.7	10000	0	51.8	10000	**2**	1.6	10000	**2**	**0.5**	10000
	200	**0**	9.5	10000	**0**	7.5	10000	**0**	100.0	10000	**0**	11.8	10000	**0**	72.5	10000	**0**	4.5	10000	**0**	**2.0**	10000
	250	**0**	7.3	10000	**0**	5.5	10000	**0**	100.0	10000	**0**	12.3	10000	**0**	73.6	10000	**0**	5.9	10000	**0**	**3.2**	10000
R1000	1000	3	4.3	725	3	7.9	2294	0	100.0	10000	3	6.0	8035	3	34.5	7217	**5**	**0.0**	**24**	**5**	**0.0**	46
	1500	1	4.0	10000	1	9.0	10000	0	100.0	10000	1	20.2	10000	1	74.3	10000	**5**	**0.0**	**97**	**5**	**0.0**	174
	2000	4	1.3	310	3	3.6	577	0	100.0	10000	1	47.3	10000	0	92.9	10000	**5**	**0.0**	44	**5**	**0.0**	**32**
	2500	**5**	**0.0**	24	4	0.6	33	0	100.0	10000	0	77.3	10000	1	75.1	10000	**5**	**0.0**	17	**5**	**0.0**	**13**
C1000	1000	4	2.0	3208	4	1.9	2220	0	100.0	10000	1	7.8	10000	1	41.1	10000	**5**	**0.0**	30	**5**	**0.0**	**16**
	1500	0	5.6	10000	0	4.0	10000	0	100.0	10000	0	15.2	10000	0	75.1	10000	**5**	**0.0**	511	**5**	**0.0**	**280**
	2000	0	6.1	10000	0	3.9	10000	0	100.0	10000	0	77.7	10000	0	78.0	10000	**3**	0.4	9556	**3**	**0.1**	**5152**
	2500	1	2.6	10000	4	0.6	**896**	0	100.0	10000	0	76.8	10000	0	87.7	10000	**5**	**0.0**	1389	**5**	**0.0**	3545
E1000	1000	0	8.3	10000	1	8.4	10000	0	100.0	10000	0	14.6	10000	0	71.0	10000	**5**	**0.0**	**840**	4	0.3	4919
	1500	0	10.2	10000	0	8.8	10000	0	100.0	10000	0	57.4	10000	0	79.9	10000	**3**	1.8	**8383**	2	**1.3**	10000
	2000	0	9.5	10000	0	7.4	10000	0	100.0	10000	0	75.1	10000	0	82.5	10000	0	3.9	10000	**1**	**1.2**	10000
	2500	**0**	7.6	10000	**0**	5.3	10000	**0**	100.0	10000	**0**	93.3	10000	**0**	90.0	10000	**0**	5.0	10000	**0**	**1.6**	10000

Table 2 provides computational results and a comparison between all models applied on the described test sets. We present the numbers of found optimal solutions, the average gaps between lower and upper bounds and the median runtimes in seconds if the method finished before reaching the time limit of 10000 seconds. As already mentioned in [6] the E instances are much harder to solve than the other instances which can easily be seen in the results especially when considering instances with large sets of different delay values and high bounds. Model SAL^{dcut} provides extremely tight LP relaxation values, see [3], where branching is hardly necessary to compute optimal integer solutions. Nevertheless, one has to consider the additional runtime for searching violated cuts which in most cases turned out to be more efficient than branching. The only advantage of the *MTZ* formulations is the independence of actual delay values and bounds. But even so it is not competitive to our approaches in almost all cases. Model *MCFL* obviously suffers from the huge amount of flow variables and therefore it is rarely applicable to the used instances. Most of the time solving model $SAL^{(dcut)}$ outperforms all other methods, but when huge sets of achievable delay values or high bounds arise, *ALF* is clearly superior. The overhead of iteratively computing small ILP models becomes worth when even LP relaxations of model $SAL^{(dcut)}$ on the full layered graph are hard to solve. Even though *ALF* is mostly slower than solving model $SAL^{(dcut)}$ it provides tight gaps and robust performance throughout all test sets. Furthermore, *ALF* consumes substantially less memory since the graphs it works on are significantly smaller than the full layered graphs, see Table 1.

7 Conclusions and Future Work

We presented two approaches to solve delay-constrained minimum tree problems based on using an appropriate layered graph. The first ILP model utilizes the special structure of this graph, mainly its acyclicity, to reduce the number of necessary variables and constraints. It provides excellent results in many cases except on instances with huge sets of possible discrete edge delay values and high bounds since the size of the layered graph heavily depends on these properties. The second approach – an algorithmic framework – tries to tackle exactly these issues. By computing lower and upper bounds for the optimal solution value on reduced layered graphs it obtains small gaps and shows robust performance throughout all test sets. Besides consuming significantly less memory it even yields tight bounds in cases where it is not possible to compute LP relaxations of the model on the full layered graph in reasonable time.

In future we try to combine heuristic methods with our adaptive layers framework to further speed up the convergence and reduce the number of necessary iterations. Also, we want to embed more sophisticated state-of-the-art solvers for the STP to additionally improve our framework. For the sake of a more comprehensive comparison we intend to re-implement the column generation and Lagrangian relaxation approach from [6]. Last but not least, we plan to adapt our framework for other optimization problems.

References

1. de Aragão, M., Uchoa, E., Werneck, R.: Dual heuristics on the exact solution of large Steiner problems. Electronic Notes in Discrete Mathematics 7, 150–153 (2001)
2. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks 1, 195–207 (1971)
3. Gouveia, L., Simonetti, L., Uchoa, E.: Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. Mathematical Programming, pp. 1–26 (2010)
4. Gouveia, L.: Using Variable Redefinition for Computing Lower Bounds for Minimum Spanning and Steiner Trees with Hop Constraints. Informs Journal on Computing 10(2), 180–188 (1998)
5. Gouveia, L.: Using hop-indexed models for constrained spanning and Steiner tree models, pp. 21–32. Kluwer Academic Publishers, Dordrecht (1999)
6. Gouveia, L., Paias, A., Sharma, D.: Modeling and Solving the Rooted Distance-Constrained Minimum Spanning Tree Problem. Computers and Operations Research 35(2), 600–613 (2008)
7. Koch, T., Martin, A.: Solving Steiner tree problems in graphs to optimality. Networks 32(3), 207–232 (1998)
8. Kompella, V.P., Pasquale, J.C., Polyzos, G.C.: Multicast routing for multimedia communication. IEEE / ACM Transactions on Networking 1(3), 286–292 (1993)
9. Leggieri, V., Haouari, M., Triki, C.: An Exact Algorithm for the Steiner Tree Problem with Delays. Electronic Notes in Discrete Mathematics 36, 223–230 (2010)
10. Ljubic, I., Gollowitzer, S.: Modelling the hop constrained connected facility location problem on layered graphs. In: Electronic Notes in Discrete Mathematics, vol. 36, pp. 207–214. Elsevier, Amsterdam (2010)
11. Ljubic, I., Weiskircher, R., Pferschy, U., Klau, G., Mutzel, P., Fischetti, M.: An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. Mathematical Programming 105(2), 427–449 (2006)
12. Manyem, P., Stallmann, M.: Some approximation results in multicasting. Tech. Rep. TR-96-03, North Carolina State University (1996)
13. Nastansky, L., Selkow, S., Stewart, N.: Cost-minimal trees in directed acyclic graphs. Mathematical Methods of Operations Research 18(1), 59–67 (1974)
14. Robins, G., Zelikovsky, A.: Improved Steiner tree approximation in graphs. In: SODA 2000: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms, pp. 770–779. Society for Industrial and Applied Mathematics (2000)
15. Ruthmair, M., Raidl, G.R.: A Kruskal-Based Heuristic for the Rooted Delay-Constrained Minimum Spanning Tree Problem. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) EUROCAST 2009. LNCS, vol. 5717, pp. 713–720. Springer, Heidelberg (2009)
16. Ruthmair, M., Raidl, G.R.: Variable Neighborhood Search and Ant Colony Optimization for the Rooted Delay-Constrained Minimum Spanning Tree Problem. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 391–400. Springer, Heidelberg (2010)
17. Xu, Y., Qu, R.: A GRASP approach for the Delay-constrained Multicast routing problem. In: Proceedings of the 4th Multidisplinary International Scheduling Conference (MISTA4), Dublin, Ireland, pp. 93–104 (2009)
18. Xu, Y., Qu, R.: A hybrid scatter search meta-heuristic for delay-constrained multicast routing problems. Applied Intelligence, 1–13 (2010)
19. Zelikovsky, A.: A series of approximation algorithms for the acyclic directed Steiner tree problem. Algorithmica 18(1), 99–110 (1997)

Jump Number of Two-Directional Orthogonal Ray Graphs

José A. Soto[1,⋆] and Claudio Telha[2]

[1] Department of Mathematics, MIT, Cambridge, MA, USA
jsoto@math.mit.edu
[2] Operation Research Center, MIT, Cambridge, MA, USA
ctelha@mit.edu

Abstract. We model *maximum cross-free matchings* and *minimum biclique covers* of *two-directional orthogonal ray graphs* (2-dorgs) as maximum independent sets and minimum hitting sets of an associated family of rectangles in the plane, respectively. We then compute the corresponding maximum independent set using linear programming and uncrossing techniques. This procedure motivates an efficient combinatorial algorithm to find a cross-free matching and a biclique cover of the same cardinality, proving the corresponding min-max relation.

We connect this min-max relation with the work of Györi [19], Lubiw [23], and Frank and Jordán [16] on seemingly unrelated problems. Our result can be seen as a non-trivial application of Frank and Jordán's Theorem.

As a direct consequence, we obtain the first polynomial algorithm for the *jump number problem* on 2-dorgs. For the subclass of convex graphs, our approach is a vast improvement over previous algorithms. Additionally, we prove that the *weighted* maximum cross-free matching problem is NP-complete for 2-dorgs and give polynomial algorithms for some subclasses.

1 Introduction

The *jump number problem* is to find a linear extension L of a poset P minimizing the number of *jumps*, that is, the number of pairs of consecutive elements in L that are incomparable in P. For bipartite posets, Chaty and Chein [8] have shown that this is equivalent to find a maximum alternating-cycle-free matching in the underlying comparability graph, which is NP-hard as shown by Pulleyblank [30].

Two related problems are the *minimum biclique cover* and the *maximum cross-free matching* problems. Given a bipartite graph $G = (A \cup B, E)$, a *biclique* is the edge set of a complete bipartite subgraph. A *biclique cover* is a family of bicliques whose union is E. Two edges e and f *cross* if there is a biclique containing both. A *cross-free matching* is a collection of pairwise non-crossing edges. Note that the maximum size of a cross-free matching is at most the minimum size of a biclique cover. An interesting question is to find classes of graphs where the two quantities coincide.

⋆ The first author was partially supported by NSF contract CCF-0829878 and by ONR grant N00014-11-1-0053.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 389–403, 2011.

For *chordal bipartite graphs*, alternating-cycle-free matchings and cross-free matchings coincide, making the jump number problem equivalent to the maximum cross-free matching problem. Müller [25] has shown that the jump number problem is NP-hard in this class. There are polynomial time algorithms for the jump number problem in important subclasses, such as bipartite permutation graphs [33,3,12], biconvex graphs [3] and convex graphs [11]. For this last class, the fastest known algorithm runs in $O(|A \cup B|^9)$ time [11]. To our knowledge, finding efficient algorithms for the jump number problem in any natural superclass of convex bipartite graphs is open in the literature.

The minimum biclique cover problem arise in many areas (e.g. biology [27], chemistry [9] and communication complexity [22]). Orlin [28] has shown that finding a minimum biclique cover of a bipartite graph is NP-hard. Müller [26] extended this result to chordal bipartite graphs. There are polynomial time algorithms for bipartite graphs that are C_4-free [26], distance hereditary graphs [26], permutation graphs [26], and domino-free graphs [1]. To our knowledge, these are the only bipartite classes for which this problem has been explicitly shown to be polynomially solvable.

Our Results. We study both the maximum cross-free matching and the minimum biclique cover problems on the class of *two-directional orthogonal ray graphs* (2-dorgs), which is a superclass of convex bipartite graphs that has recently been studied by many authors [29,32]. Our main result is that the size of a maximum cross-free matching is equal to the size of a minimum biclique cover in 2-dorgs, and both objects are polynomially computable. A key tool for this result is a new *geometrical reformulation* of the previous problems as the *maximum independent set* and the *minimum hitting set* of an associated collection of rectangles in the plane respectively. Our reformulation is reminiscent of the one already observed by Ceroi [5], between the jump number of a two-dimensional poset and the maximum *weighted* independent set of a collection of rectangles in the plane. Using this geometric representation, we give a linear program formulation for the maximum cross-free matching. Even though the associated polytope is not integral, we show how to find an optimal integral vertex by using an uncrossing procedure. We can also find this vertex by solving a related linear program having only integral optimal extreme points. We are further able to devise an efficient combinatorial algorithm that computes simultaneously a maximum cross-free matching and a minimum biclique cover of the same cardinality in a 2-dorg. We also study a weighted version of the maximum cross-free matching problem, which is equivalent to the *weighted jump number* studied by Ceroi [6] and show that this problem is NP-hard for 2-dorgs.

We explore the relation between our main result and the following, apparently unrelated, pairs of combinatorial problems where a min-max result exists: the *minimum rectangle cover* and the *maximum antirectangle* of an orthogonal biconvex board, studied by Chaiken et al. [7]; the *minimum base of a family of intervals* and the *maximum independent set of point-interval pairs*, studied by Győri [19] and Lubiw [23]; and finally, the *minimum edge-cover* and the *maximum half-disjoint family of set-pairs*, studied by Frank and Jordán [16]. Our

result can be seen both as a generalization of Györi's result and as a non-trivial application of Frank and Jordán's Theorem.

Additionally, for special subclasses of 2-dorgs, we give new results: (1) We give new efficient algorithms to solve the maximum weight cross-free matching in bipartite permutation graphs. (2) For convex graphs, we show how to use algorithmic versions of Györi's result [18,21] to compute a maximum cross-free matching and minimum biclique cover in $O(n^2)$ time and how to use a result by Lubiw [23] to compute a maximum weight cross-free matching in $O(n^3)$ time.

2 Preliminaries

Notation. The *rectangles* we consider in this paper are closed sets in the plane (possibly not full dimensional). Their sides are parallel to the x and y axes and their vertices are points in $\boldsymbol{Z}^2$. For a set $S \in \boldsymbol{Z}^2$, we denote by S_x the projection of S onto the x-axis. If $S = \{p\}$ is a singleton, we write S_x simply as p_x. For sets $S, S' \in \boldsymbol{Z}^2$, we write $S_x < S'_x$ if the projection S_x is to the left of the projection S'_x, that is, if $p_x < p'_x$ for all $p \in S$, $p' \in S'$. We extend this convention to $S_x > S'_x, S_x \leq S'_x, S_x \geq S'_x$ and to the projections onto the y-axis as well.

Two special sets of points $A, B \subseteq \boldsymbol{Z}^2$, not necessarily disjoint, represent the vertices of the bipartite graphs we consider in this paper. We use a and b to denote points of A and B respectively. For $a_x \leq b_x$ and $a_y \leq b_y$, we use $\Gamma(a, b)$ to the denote the rectangle with bottom-left corner a and top-right corner b. Let $\mathcal{R}(A, B)$ (or simply $\mathcal{R}$) be the set of rectangles with bottom-left corner in A and top-right corner in B. This is, $\mathcal{R} = \{\Gamma(a, b) \colon a \in A, b \in B, a_x \leq b_x, a_y \leq b_y\}$. The subset of inclusion-wise minimal rectangles of $\mathcal{R}$ is denoted by $\mathcal{R}_\downarrow$. Abusing notation, we denote by $G = (A \cup B, \mathcal{R})$ the bipartite graph with bipartition A and B, where there is an edge between $a \in A$ and $b \in B$ if and only if $\Gamma(a, b) \in \mathcal{R}$. Finally, for a given rectangle $R \in \mathcal{R}$, we denote by $A(R)$ (resp. $B(R)$) the bottom-left (resp. top-right) corner of the rectangle R.

We claim that the class of graphs arising from the previous construction is equivalent to the class of *two-directional orthogonal ray graphs* (2-dorgs) recently considered by Shrestha et al. [32]. A 2-dorg is a bipartite graph on $A \cup B$ where each vertex v is associated to a point $(v_x, v_y) \in \boldsymbol{Z}^2$, so that $a \in A$ and $b \in B$ are connected if and only the rays $[a_x, \infty) \times \{a_y\}$ and $\{b_x\} \times (-\infty, b_y]$ intersect each other. Since this condition is equivalent to $\Gamma(a, b) \in \mathcal{R}(A, B)$, the claim follows.

It is an easy exercise to prove that every 2-dorg admits a geometric representation where no two points of $A \cup B$ are in the same horizontal or vertical line, and furthermore, $A \cup B$ are points of the grid $[n]^2 = \{1, \ldots, n\} \times \{1, \ldots, n\}$, where $n = |A \cup B|$. When these conditions hold, we say that $G = (A \cup B, \mathcal{R})$ is in *rook representation*. For the rest of the paper, we consider a 2-dorg as being *both the graph and its geometric representation*.

Graph Definitions. We recall the following definitions of a nested family of bipartite graph classes. We keep the notation $G = (A \cup B, \mathcal{R})$ since we can show they are 2-dorgs.

A *bipartite permutation graph* is the comparability graph of a two dimensional poset of height 2, where A is the set of minimal elements and B is the complement of this set. A *two dimensional poset* is a collection of points in $\boldsymbol{Z}^2$ with the relation $p \leq_{\boldsymbol{Z}^2} q$ if $p_x \leq q_x$ and $p_y \leq q_y$.

In a *convex graph*, there is a labeling for $A = \{a_1, \ldots, a_k\}$ so that the neighborhood of each $b \in B$ is a set of consecutive elements of A. In a *biconvex graph*, there is also a labeling for $B = \{b_1, \ldots, b_l\}$ so that the neighborhood of each $a \in A$ is consecutive in B.

In an *interval bigraph*, each vertex $v \in A \cup B$ is associated to a real closed interval I_v (w.l.o.g. with integral extremes) so that $a \in A$ and $b \in B$ are adjacent if and only if $I_a \cap I_b \neq \emptyset$.

It is known that bipartite permutation $\subset$ biconvex $\subset$ convex $\subset$ interval bigraph $\subset$ 2-dorg and that all inclusions are strict [4,32]. We give a simple proof of the last inclusion using our geometrical interpretation of 2-dorgs. Let G be an interval bigraph with parts A and B. For $a \in A$ with interval $I_a = [s, t]$ and $b \in B$ with interval $I_b = [s', t']$, we identify a with the point $(s, -t) \in \boldsymbol{Z}^2$ and b with the point $(t', -s') \in \boldsymbol{Z}^2$. By definition, ab is an edge of G if and only if $[s, t] \cap [s', t'] \neq \emptyset$, or equivalently if $s \leq t'$ and $-t \leq -s'$. This shows that $\Gamma(a, b)$ is in $\mathcal{R}(A, B)$. Intuitively, the previous assignment maps A (resp. B) to points weakly below (resp. weakly above) the diagonal line $y = -x$ in such a way that their horizontal and vertical projections onto this line define the corresponding intervals. We illustrate this construction in Fig. 1.

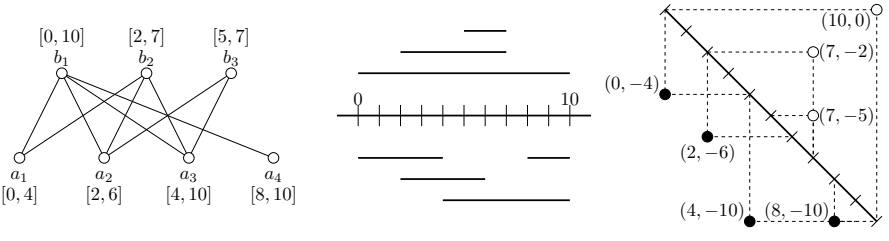


Fig. 1. An interval bigraph, its interval representation and a geometric representation as 2-dorg

Geometric Interpretation. Given a bipartite graph $G = (V, E)$, a *biclique* is the edge set of a complete bipartite subgraph of G. A *biclique cover* is a family of bicliques whose union is E. Two distinct edges $e = ab$ and $f = a'b'$ *cross* if there is a biclique containing both e and f. This holds if and only if ab' and $a'b$ are also edges of the graph. In particular, two edges incident to the same vertex cross. A *cross-free matching* is a collection of edges pairwise non-crossing. We denote the maximum size of a cross-free matching by $\alpha^*(G)$ and the minimum size of a biclique cover by $\kappa^*(G)$. For the last quantity, we can restrict to biclique covers using maximal bicliques.

Theorem 1. *Let $G = (A \cup B, \mathcal{R})$ be a 2-dorg. Two edges R and R' cross if and only if R and R' intersect as rectangles.*

Proof. Let $R = \Gamma(a, b)$ and $R' = \Gamma(a', b')$ be distinct edges in $\mathcal{R}$. Both edges cross if and only if $\Gamma(a, b')$ and $\Gamma(a', b)$ are also in $\mathcal{R}$. This is equivalent to $\max(a_x, a'_x) \leq \min(b_x, b'_x)$ and $\max(a_y, a'_y) \leq \min(b_y, b'_y)$. From here, if R and R' cross, then the point $p = (\max(a_x, a'_x), \max(a_y, a'_y))$ is in the intersection of R and R' as rectangles. Conversely, if there is a point $p \in R \cap R'$, then $\max(a_x, a'_x) \leq p_x \leq \min(b_x, b'_x)$ and $\max(a_y, a'_y) \leq p_y \leq \min(b_y, b'_y)$, and so, R and R' cross.

It is natural to study the following problems. Given a collection $\mathcal{C}$ of rectangles, an *independent set* is a family of pairwise disjoint rectangles in $\mathcal{C}$ and a *hitting set* is a set of points such that every rectangle in $\mathcal{C}$ contains at least one point of this set. We denote by $\mathrm{mis}(\mathcal{C})$ and $\mathrm{mhs}(\mathcal{C})$ the sizes of a *maximum independent set of rectangles* in $\mathcal{C}$ and a *minimum hitting set* for $\mathcal{C}$ respectively.

Consider the intersection graph $\mathcal{I}(\mathcal{C}) = (\mathcal{C}, \{RS : R \cap S \neq \emptyset\})$. Naturally, independent sets of $\mathcal{C}$ correspond to stable sets in $\mathcal{I}(\mathcal{C})$. On the other hand, since the family $\mathcal{C}$ has the *Helly property*,[1] we can assign to every clique in $\mathcal{I}(\mathcal{C})$ a unique *witness point*, defined as the leftmost and lowest point contained in all rectangles of the clique. Since different maximal cliques have different witness points, it is easy to prove that $\mathcal{C}$ admits a minimum hitting set consisting only of witness points of maximal cliques. In particular, $\mathrm{mhs}(\mathcal{C})$ is equal to the minimum size of a *clique-cover* of $\mathcal{I}(\mathcal{C})$.

For both problems defined above we can restrict ourselves to the family $\mathcal{C}_\downarrow$ of inclusion-wise minimal rectangles in $\mathcal{C}$, since any maximum independent set in $\mathcal{C}_\downarrow$ is also maximum in $\mathcal{C}$ and any minimum hitting set for $\mathcal{C}_\downarrow$ is also minimum for $\mathcal{C}$. Using Theorem 1 we conclude the following.

Theorem 2. *For a 2-dorg $G = (A \cup B, \mathcal{R})$, the cross-free matchings of G correspond to the independent sets of $\mathcal{R}$ and the biclique covers of G using maximal bicliques correspond to the hitting sets for $\mathcal{R}$ using witness points of maximal cliques (and to clique-covers in $\mathcal{I}(\mathcal{R})$ using maximal cliques). In particular, $\alpha^*(G) = \mathrm{mis}(\mathcal{R}) = \mathrm{mis}(\mathcal{R}_\downarrow)$ and $\kappa^*(G) = \mathrm{mhs}(\mathcal{R}) = \mathrm{mhs}(\mathcal{R}_\downarrow)$.*

3 A Linear Programming Approach

Consider the integer program formulation for the maximum independent set of a collection of rectangles $\mathcal{C}$ with vertices on the grid $[n]^2$:

$$\mathrm{mis}(\mathcal{C}) = \max\left\{ \sum_{R \in \mathcal{C}} x_R \colon \sum_{R\colon\, q \in R} x_R \leq 1, q \in [n]^2; x \in \{0,1\}^{\mathcal{C}} \right\} .$$

Let $\mathrm{P}(\mathcal{C})$ be the polytope $\{x \in \boldsymbol{R}^{\mathcal{C}} \colon \sum_{R\colon\, q \in R} x_R \leq 1, q \in [n]^2; x \geq 0\}$ and the relaxation of $\mathrm{mis}(\mathcal{C})$ given by $\mathrm{LP}(\mathcal{C}) = \max\{\sum_{R \in \mathcal{C}} x_R \colon x \in \mathrm{P}(\mathcal{C})\}$.

[1] If a collection of rectangles intersect, then all of them share a rectangular region in the plane.

Let $G = (A \cup B, \mathcal{R})$ be a 2-dorg with all its vertices in the grid $[n]^2$. If $\mathrm{LP}(\mathcal{R})$ (or $\mathrm{LP}(\mathcal{R}_\downarrow)$) has an integral solution then, by Theorem 2, this solution induces a maximum cross-free matching of G. This happens, for example, when $\mathrm{P}(\mathcal{R})$ or $\mathrm{P}(\mathcal{R}_\downarrow)$ is an integral polytope.

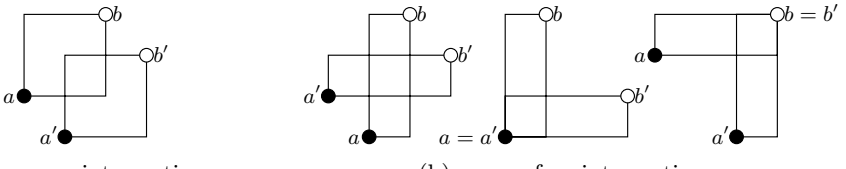
Theorem 3. *Given a family of rectangles $\mathcal{C}$ with vertices in $[n]^2$, the polytope $\mathrm{P}(\mathcal{C})$ is integral if and only if the intersection graph $\mathcal{I}(\mathcal{C})$ is* perfect.

Theorem 3 follows from the fact that $\mathrm{P}(\mathcal{C})$ is the *clique-constrained stable set polytope* of the intersection graph $\mathcal{I}(\mathcal{C})$, usually denoted by $\mathrm{QSTAB}(\mathcal{I}(\mathcal{C}))$. A classical result on perfect graphs (See, e.g. [31]) establishes that a graph H is perfect if and only if $\mathrm{QSTAB}(H)$ is integral.

If $G = (A \cup B, \mathcal{R})$ is a 2-dorg such that $\mathcal{I}(\mathcal{R})$ (or $\mathcal{I}(\mathcal{R}_\downarrow)$) is perfect, then $\alpha^*(G) = \kappa^*(G)$ and solving the linear program $\mathrm{LP}(\mathcal{R})$ (or $\mathrm{LP}(\mathcal{R}_\downarrow)$) gives a polynomial time algorithm for finding a maximum cross-free matching. Since there are also polynomial time algorithms to find minimum clique-covers of perfect graphs, we can also obtain a minimum biclique cover of G.

For bipartite permutation graphs $G = (A \cup B, \mathcal{R})$, the intersection graph $\mathcal{I}(\mathcal{R}) \cong (\mathcal{R}, \{ef : e \text{ and } f \text{ cross}\})$ is known to be perfect since it is both weakly chordal [26] and co-comparability [3]. By the previous discussion we can find a maximum cross-free matching and minimum biclique cover for these graphs in polynomial time. Using the structure of bipartite permutation graphs, linear-time algorithms have been developed for both problems [33,3,12]. For biconvex graphs $G = (A \cup B, \mathcal{R})$, the graph $\mathcal{I}(\mathcal{R})$ is not necessarily perfect, but we can show that there is a geometric representation $(A' \cup B', \mathcal{R}')$ for which $\mathcal{I}(\mathcal{R}'_\downarrow)$ is perfect and that this property does not extend to convex graphs. We defer the proof of these facts for the full version of this paper.

Even if the intersection graph of the rectangles of a 2-dorg is not perfect, we can still find a maximum cross-free matching. Let $G = (A \cup B, \mathcal{R})$ be a 2-dorg in *rook representation*. Since the rectangles in $\mathcal{R}_\downarrow$ are inclusion-wise minimal, no rectangle $R = \Gamma(a, b)$ in $\mathcal{R}_\downarrow$ contains a third point in $(A \cup B) \setminus \{a, b\}$. Therefore, there are only four ways in which a pair of rectangles can intersect. They are depicted in Fig. 2. We say that two intersecting rectangles have *corner-intersection* if all the vertices involved are distinct and each rectangle contain a corner of the other. If this does not happen, we say that the intersection is *corner-free*. A *corner-free-intersection* (c.f.i.) family is a collection of inclusion-wise minimal rectangles having no corner-intersections.



(a) corner-intersection (b) corner-free intersections

Fig. 2. The ways two rectangles in $\mathcal{R}_\downarrow$ can intersect each other

Let z^* be the optimal value of $\mathrm{LP}(\mathcal{R}_\downarrow)$ and for every rectangle $R \in \mathcal{R}_\downarrow$, let $\mu(R) > 0$ be its area.[2] Let $\bar{x}$ be an *optimal extreme point* of the following linear program

$$\mathrm{LP}(z^*, \mathcal{R}_\downarrow) = \min\Big\{ \sum_{R\in\mathcal{R}_\downarrow} \mu(R)x_R \colon \sum_{R\in\mathcal{R}_\downarrow} x_R = z^* \text{ and } x \in \mathrm{P}(\mathcal{R}_\downarrow)\Big\} \ .$$

Note that $\bar{x}$ is a solution to $\mathrm{LP}(\mathcal{R}_\downarrow)$ minimizing its total *weighted area* and it is also an optimal extreme point of $\max\{\sum_{R\in\mathcal{R}_\downarrow}(1-\varepsilon\mu(R))x_r \colon x \in \mathrm{P}(\mathcal{R}_\downarrow)\}$ for a small value of ε.

Theorem 4. *The point $\bar{x}$ is an integral point.*

Proof. Suppose that $\bar{x}$ is not integral. Let $\overline{\mathcal{R}} = \{R \in \mathcal{R}_\downarrow \colon \bar{x}_R > 0\}$ be the set of rectangles in the support of $\bar{x}$. We claim that $\overline{\mathcal{R}}$ is a c.f.i. family. Assume for sake of contradiction that $R = \Gamma(a,b)$ and $R' = \Gamma(a',b')$ are rectangles in $\overline{\mathcal{R}}$ having corner-intersection as in Fig. 2a. We apply the following *uncrossing procedure*: let $\varepsilon = \min(\bar{x}_R, \bar{x}_{R'}) > 0$ and consider the rectangles $S = \Gamma(a,b')$ and $S' = \Gamma(a',b)$ in $\mathcal{R}_\downarrow$. Consider the vector $\tilde{x}$ obtained from $\bar{x}$ by decreasing x_R and $x_{R'}$ by ε and increasing by the same amount x_S and $x_{S'}$. It is easy to see that $\tilde{x}$ is a feasible solution of $\mathrm{LP}(z^*, \mathcal{R}_\downarrow)$ with strictly smaller weighted area than $\bar{x}$, contradicting its optimality.

Now we prove that the intersection graph $\mathcal{I}(\overline{\mathcal{R}})$ is a comparability graph, and therefore it is perfect. Consider the following partial order in $\overline{\mathcal{R}}$:

$$R \preceq R' \text{ if and only if } R_y \subseteq R'_y \text{ and } R_x \supseteq R'_x \ , \tag{1}$$

Since $\overline{\mathcal{R}}$ is a c.f.i. family, it follows that every pair of intersecting rectangles in $\overline{\mathcal{R}}$ are comparable by $\preceq$. The converse trivially holds. Therefore, $\mathcal{I}(\overline{\mathcal{R}})$ is the comparability graph of $(\overline{\mathcal{R}}, \preceq)$.

Using Theorem 3 we show that $\mathrm{P}(\overline{\mathcal{R}})$ is an integral polytope. Consider the set $F = \mathrm{P}(\overline{\mathcal{R}}) \cap \{\sum_{R\in\overline{\mathcal{R}}} x_R = z^*, \sum_{R\in\overline{\mathcal{R}}} \mu(R)x_R = \sum_{R\in\overline{\mathcal{R}}} \mu(R)\bar{x}_R\}$. This set is a face of $\mathrm{P}(\overline{\mathcal{R}})$ containing only optimum solutions of $\mathrm{LP}(z^*, \mathcal{R}_\downarrow)$. To conclude the proof of the theorem we show that $\bar{x}$ is a vertex of F, and therefore, a vertex of $\mathrm{P}(\overline{\mathcal{R}})$. If $\bar{x}$ is not a vertex of F, then we can find two different points in F such that $\bar{x}$ is a convex combination of those points. This means that $\bar{x}$ is a convex combination of different optimum solutions of $\mathrm{LP}(z^*, \mathcal{R}_\downarrow)$, contradicting the choice of $\bar{x}$.

Using Theorem 4 we can find a maximum cross-free matching of a 2-dorg G in rook representation as follows: solve the linear program $\mathrm{LP}(\mathcal{R}_\downarrow)$ and find an optimal extreme point of $\mathrm{LP}(z^*, \mathcal{R}_\downarrow)$. Moreover, since 2-dorgs are *chordal-bipartite graphs* [32], for which the maximum cross-free matching and the jump number problems are equivalent, we conclude the following.

[2] For our discussion, the *area* or a rectangle $R = \Gamma(a,b)$ is defined as $(b_x - a_x)(b_y - a_y)$. However, our techniques also works if we define the *area* of a rectangle as the number of grid points it contains.

Theorem 5. *For a 2-dorg, the maximum cross-free matching (and equivalently, the jump number) can be computed in polynomial time.*

For any family of rectangles $\mathcal{C}$ with vertices in $[n]^2$, consider the linear program $\mathrm{DP}(\mathcal{C}) = \min\{\sum_{q\in[n]^2} y_q \colon \sum_{q\in R} y_q \geq 1, R \in \mathcal{C}; y \geq 0\}$. The integer feasible solutions of $\mathrm{DP}(\mathcal{C})$ correspond exactly to hitting sets for $\mathcal{C}$ that are contained in the grid $[n]^2$. Since there are minimum hitting sets for $\mathcal{C}$ of this type, we conclude that $\mathrm{DP}(\mathcal{C})$ is a linear program relaxation for $\mathrm{mhs}(\mathcal{C})$.

The programs $\mathrm{LP}(\mathcal{C})$ and $\mathrm{DP}(\mathcal{C})$ are dual to each other. Hence, they have a common optimum value $z^*(\mathcal{C})$ and $\mathrm{mis}(\mathcal{C}) \leq z^*(\mathcal{C}) \leq \mathrm{mhs}(\mathcal{C})$. In particular, for the family $\mathcal{R}_\downarrow$ of inclusion-wise minimal rectangles coming from a 2-dorg G in rook representation, we have

$$\alpha^*(G) \leq z^*(\mathcal{R}_\downarrow) \leq \kappa^*(G) \ . \tag{2}$$

In this section we have shown not only that the first inequality in (2) is an equality, but also that an integer optimal solution for $\mathrm{LP}(\mathcal{R}_\downarrow)$ can be found by optimizing an easy to describe linear function over the optimal face of this linear program. It would be interesting to show a similar result for $\mathrm{DP}(\mathcal{R}_\downarrow)$, since as a consequence of the combinatorial algorithm we give in Sect. 4, this polytope also admits an integer optimal solution.

4 A Combinatorial Algorithm

In this section we give a combinatorial algorithm that computes simultaneously a maximum cross-free matching and a minimum biclique cover of a 2-dorg. Our procedure is based on the algorithmic proof of Győri's min-max result of intervals [19] given by Frank [15]. In order to keep the discussion self-contained, we do not rely on previous results. The relation of our approach to other previous work will be explored in Sect. 5.

In Sect. 3 we have shown that a maximum cross-free matching of a 2-dorg $G = (A \cup B, \mathcal{R})$ can be obtained from a maximum independent set of the c.f.i. family $\overline{\mathcal{R}}$. In what follows, we show that this result also holds if we replace $\overline{\mathcal{R}}$ by certain *maximal greedy* c.f.i. subfamily of $\mathcal{R}_\downarrow$.

Given a 2-dorg $G = (A \cup B, \mathcal{R})$ in rook representation, we say that a rectangle $R \in \mathcal{R}_\downarrow$ appears before a rectangle $S \in \mathcal{R}_\downarrow$ in *right-top order* if we either have $A(R)_x < A(S)_x$, or if $A(R)_x = A(S)_x$ and $B(R)_y < B(S)_y$. This defines a total order on $\mathcal{R}_\downarrow$. Construct a family $\mathcal{K}$ by processing the rectangles in $\mathcal{R}_\downarrow$ in right-top order and adding only those that keep $\mathcal{K}$ corner-free.

Since $\mathcal{I}(\mathcal{K})$ is the comparability graph of $(\mathcal{K}, \preceq)$, where $\preceq$ is as in (1), the size of a maximum independent set $\mathcal{R}_0$ of $\mathcal{K}$ is equal to the size of a minimum hitting set H_0 for $\mathcal{K}$. We can find optimal solutions of these problems by computing a maximum antichain and a minimum chain-cover of the poset $(\mathcal{K}, \preceq)$, using any polynomial time algorithm for Dilworth's chain-partitioning problem (See, e.g. [13]). We modify H_0 to obtain a set of points H^* of the same size hitting $\mathcal{R}$.

An *admissible flip* of a hitting set H for $\mathcal{K}$ is an ordered pair of points p and q in H with $p_x < q_x$ and $p_y < q_y$, such that the set $H \setminus \{p, q\} \cup \{(p_x, q_y), (q_x, p_y)\}$ obtained by *flipping p and q* is still a hitting set for $\mathcal{K}$. Construct H^* from H_0 by flipping admissible flips while this is possible. Note that flipping two points reduces the potential $\psi(H) = \sum_{p \in H} p_x p_y$ by at least one unit. Since ψ is positive and $\psi(H_0) \leq |H_0| n^2 \leq n^3$, the previous construction can be done efficiently.

Lemma 1. *H^* is a hitting set for $\mathcal{R}_\downarrow$, and therefore, a hitting set for $\mathcal{R}$.*

Proof. Suppose this is not the case. Let $R = \Gamma(a, b)$ be the *last* rectangle of $\mathcal{R}_\downarrow \setminus \mathcal{K}$ not hit by H^*, in right-top order. Let also $R' = \Gamma(a', b')$ be the *first* rectangle of $\mathcal{K}$ having corner-intersection with R, in right-top order. We have $a'_x < a_x < b'_x < b_x$ and $b'_y > b_y > a'_y > a_y$ (See Fig. 3). In particular, the rectangles $S = \Gamma(a', b)$ and $T = \Gamma(a, b')$ are in $\mathcal{R}$. They are also inclusion-wise minimal, otherwise there would be a point $v \in A \cup B \setminus \{a, a', b, b'\}$ in $S \cup T \subseteq R \cup R'$, contradicting the minimality of R or R'. Since T appears after R in right-top order, it is hit by a point $q \in H^*$.

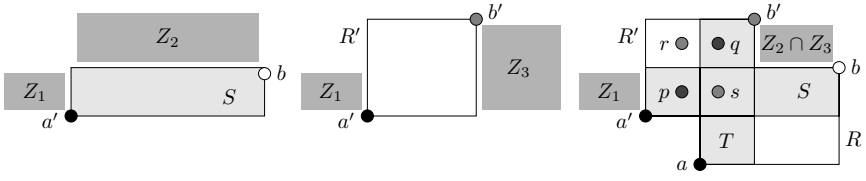


Fig. 3. Positions of rectangles and zones defined in the proof of Lemma 1

We claim that the rectangle S is in $\mathcal{K}$. If this is not the case, then there is a rectangle $R'' = \Gamma(a'', b'') \in \mathcal{K}$ appearing before S and having corner-intersection with S. In particular corner a'' lies in $Z_1 = (-\infty, a'_x - 1] \times [a'_y + 1, b_y - 1]$, and corner b'' lies in the zone $Z_2 = [a'_x + 1, b_x - 1] \times [b_y + 1, \infty)$ as shown on the left of Fig. 3. Note that the top-left corner (a'_x, b_y) of rectangle S is in both R' and R''. Since both are in $\mathcal{K}$, the rectangles R' and R'' have corner-free intersection. Using this last fact, and that $a'' \in Z_1$ we conclude that b'' is either equal to b' or it lies in the zone $Z_3 = [b'_x + 1, \infty) \times [a'_y + 1, b_y - 1]$. See the center of Fig. 3.

Since $a'' \in Z_1$ and $b'' \in \{b'\} \cup (Z_2 \cap Z_3)$, we conclude that R and R'' have corner-intersection, contradicting the choice of R' since R'' appears before R' in right-top order. This proves the claim and, since $S \in \mathcal{K}$, it must be hit by a point $p \in H^*$.

Now we show that p and q are an admissible flip for H^*, contradicting the construction of H^*. Since $p \in S \setminus R$ and $q \in T \setminus R$ we have $p_x < a_x \leq q_x$ and $p_y \leq b_y < q_y$. Let $r = (p_x, q_y)$, $s = (q_x, p_y)$ and suppose there is a rectangle $U \in \mathcal{K}$ hit by H^* but not by $(H^* \setminus \{p, q\}) \cup \{r, s\}$. If the rectangle U is hit by p (but not by r or s), then its upper-right corner $B(U)$ must be in the region $[p_x, q_x - 1] \times [p_y, q_y - 1]$. In particular, $B(U) \in R' \setminus \{a', b'\}$, contradicting the inclusion-wise minimality of R'. If on the other hand U is hit by q (but not by r or s), then its bottom-left corner $A(U)$ must be in $[p_x + 1, q_x] \times [p_y + 1, q_y]$.

As before, this means that $A(U) \in R' \setminus \{a', b'\}$, contradicting the inclusion-wise minimality of R'. Therefore, p and q are an admissible flip, concluding the proof of the lemma.

Using Lemma 1, we can find a maximum cross-free matching and a minimum biclique cover of a 2-dorg $G = (A \cup B, \mathcal{R})$ in rook representation using the following algorithm: Compute the c.f.i. greedy family $\mathcal{K}$ as above. Use an algorithm for Dilworth's chain-partitioning problem to compute a maximum independent set $\mathcal{R}_0$ and a minimum hitting set H_0 for $\mathcal{K}$. Finally, compute the set H^* as described above. Since H^* is a hitting set for $\mathcal{R}$, and $\mathcal{R}_0$ is an independent set of $\mathcal{R}$ with $|H^*| = |H_0| = |\mathcal{R}_0|$, we conclude they are both optima; therefore, they induce a maximum cross-free matching and a minimum biclique cover for G.

Theorem 6. *The previous algorithm computes a maximum cross-free matching and a minimum biclique cover of the same size for any 2-dorg G in polynomial time. In particular, $\alpha^*(G) = \kappa^*(G)$.*

Dilworth's chain-partitioning problem on $(\mathcal{K}, \preceq)$ can be solved by finding a maximum matching on a bipartite graph with $2|\mathcal{K}|$ vertices [13]. This task can be done in $O(|\mathcal{K}|^{2.5})$ time using Hopcroft-Karp algorithm [20], or in $O(|\mathcal{K}|^{\omega})$ randomized time, where ω is the exponent for matrix multiplication, using an algorithm by Mucha and Sankowski [24]. A naive implementation of our algorithm using these results runs in $O(n^7)$. In the full version of the paper, we give faster implementations running in $\widetilde{O}(n^{2.5})$ time and $\widetilde{O}(n^{\omega})$ randomized time.

5 Discussion

In this section, we study the connection of maximum cross-free matchings and minimum biclique covers with other combinatorial problems in the literature. Chaiken et al. [7] have studied the problem of covering a biconvex board (an orthogonal polygon horizontally and vertically convex) with orthogonal rectangles included in this board. They have shown that the *minimum size of a rectangle cover* is equal to the *maximum size of an antirectangle* (a set of points in the board such that no two of them are covered by a rectangle included in the board). Györi [19] has shown that the same result holds even when the board is only vertically convex, as a corollary of a min-max relation we describe.

A collection of point-interval pairs (p_j, I_j), with $p_j \in I_j$ is *independent* if $p_j \notin I_k$ or $p_k \notin I_j$ for $k \neq j$. A family of intervals is a *base* for another family if every interval of the latter is a union of intervals in the former. Györi establishes that the size of the *largest independent set of point-interval pairs* with intervals in a family F is equal to the size of the *minimum base of* F, where this base is not restricted to be a subset of F. The corresponding optimizers can be found using an algorithm of Franzblau and Kleitman [18] (or an implementation by Knuth [21]) in $O(k^2)$ time where k is the number of different endpoints of F. These algorithms can also be used to compute a maximum antirectangle and a minimum rectangle cover of a convex board.

Every biconvex graph G admits a biadjacency matrix where the set of *ones* form a biconvex board $\mathcal{B}(G)$. Brandstädt [3] notes that finding a maximum cross-free matching (which he called alternating-C_4-free matching) in a biconvex graph G is equivalent to finding a maximum antirectangle in $\mathcal{B}(G)$. By the previous discussion, this can be done in polynomial time. Here we observe that the maximal rectangles in $\mathcal{B}(G)$ correspond exactly to the maximal bicliques of G, showing that the minimum biclique cover of G can be obtained with the same algorithm. Surprisingly, to the authors knowledge, this observation has not been used in the literature concerning the minimum biclique cover problem, where this problem remains open for biconvex graphs.

Similar to the previous case, every convex graph $G = (A \cup B, \mathcal{R})$ admits a biadjacency matrix where the set of *ones* forms a vertically convex board $\mathcal{B}(G)$. However, antirectangles and rectangle covers of $\mathcal{B}(G)$ are no longer in correspondence with cross-free matchings and biclique covers of G. Nevertheless, we can still make a connection to Györi's result on intervals as follows. Every element of B can be seen as an *interval* of elements in A. Cross-free matchings in G correspond then to independent families of point-intervals, with intervals in B. Similarly, since every maximal biclique of G is defined by taking some interval I of elements in A (where I does not necessarily correspond to some $b \in B$), as $\{(a, b) \colon a \in I, b \supseteq I\}$, we obtain that minimal biclique covers of G (using maximal bicliques) correspond to minimum generating families of B. We remark that this connection has not been noted before in the literature. As a corollary of this discussion we have the following.

Corollary 1. *The maximum cross-free matching and minimum biclique cover of convex and biconvex graphs can be computed in $O(n^2)$ time using Knuth's [21] implementation of the algorithm of Franzblau and Kleitman [18].*

In a seminal paper, Frank and Jordán [16] extend Györi's result to *set-pairs*. We briefly describe a particular case of their result that concerns us. A collection of pairs of sets $\{(S_i, T_i)\}$ is *half-disjoint* if for every $i \neq j$, $S_i \cap S_j$ or $T_i \cap T_j$ is empty. A *directed-edge* (s, t) covers a set-pair (S, T) if $s \in S$ and $t \in T$. A family $\mathcal{S}$ of set-pairs is *crossing* if whenever (S, T) and (S', T') are in $\mathcal{S}$, so are $(S \cap T, S' \cup T')$ and $(S \cup T, S' \cap T')$. Frank and Jordán prove that for every crossing family $\mathcal{S}$, the *maximum size of a half-disjoint subfamily* is equal to the *minimum size of a collection of directed-edges covering* $\mathcal{S}$. They also give a linear programming based algorithm to compute both optimizers. Later, combinatorial algorithms for this result were also given (e.g. [2]). See Végh's Ph.D. thesis [34] for related references.

Theorems 5 and 6 can be seen as non-trivial applications of Frank and Jordán's result. Given a 2-dorg $G = (A \cup B, \mathcal{R})$, consider the family of set-pairs $\mathcal{S} = \{(R_x, R_y) \colon R \in \mathcal{R}_\downarrow\}$. It is easy to check that this family is crossing, that half-disjoint families of $\mathcal{S}$ correspond to independent sets in $\mathcal{R}_\downarrow$ and that coverings of $\mathcal{S}$ by directed-edges correspond to hitting sets for $\mathcal{R}_\downarrow$. We remark that this reduction relies heavily on the geometric interpretation of 2-dorgs we have presented in this paper, and that our proofs are self-contained and simpler than the ones used to prove the broader result of Frank and Jordán.

We want to point out that the combinatorial algorithm presented in Sect. 4 extends the algorithm given by Frank [15] as an alternative proof of Győry's result on intervals. But at the same time it comprises the same algorithmic ideas from other applications of Frank and Jordan's result, such as the algorithm of Frank and Végh [17] for connectivity augmentation. Our description is tailored to the instances considered in this paper, leading to a simpler description of the algorithm and a simpler running time analysis.

6 The Maximum Weight Cross-Free Matching Problem

We now consider the problem of finding the *maximum weight cross-free matching* of 2-dorgs $G = (A \cup B, \mathcal{R})$ with non-negative weights $\{w_R\}_{R \in \mathcal{R}}$. This problem is equivalent to the maximum weight jump number of (G, w) defined by Ceroi [6] and to the maximum weight independent set of $(\mathcal{R}, w)$.

The maximum weight cross-free matching is NP-hard for 2-dorgs, even if the weights are zero or one. To see this, we reduce from maximum independent set of rectangles (MISR), which is NP-hard even if the vertices of the rectangles are all distinct [14]. Given an instance $\mathcal{I}$ of MISR with the previous property, let A (resp. B) be the set of lower-left (resp. upper-right) corners of rectangles in $\mathcal{I}$. Note that the 2-dorg $G = (A \cup B, \mathcal{R})$ satisfies $\mathcal{I} \subseteq \mathcal{R}$ so we can find the MISR of $\mathcal{I}$ by finding the maximum weight cross-free matching in G, where we give a weight of one to each $R \in \mathcal{I}$, and a weight of zero to every other rectangle.

Theorem 7. *The maximum weight cross-free matching problem is* NP*-hard for 2-dorgs.*

We now provide an efficient algorithm for the maximum weight cross-free matching of bipartite permutation graphs. We use the natural 2-dorg representation $G = (A \cup B, \mathcal{R})$ arising from the definition of this class. We can solve the maximum weight independent set of $\mathcal{R}$ in $O(n^2)$ time using the fact that the complement of the intersection graph $\mathcal{I}(\mathcal{R})$ is a comparability graph. To see this, let us write $R \searrow S$ if R and S are disjoint and either $R_x < S_x$ or $R_y > S_y$ holds. It is not hard to verify that $D = (\mathcal{R}, \searrow)$ is a partial order whose comparability graph is the complement of $\mathcal{I}(\mathcal{R})$, and that maximum weight cross-free matchings in G corresponds to maximum weight paths in the digraph D, using w as a weight function on the vertex set $\mathcal{R}$. Since D has $|\mathcal{R}|$ vertices and $O(|\mathcal{R}|^2)$ arcs this optimal path Q^* can be found in $O(|\mathcal{R}|^2)$ time [10].

We can find Q^* faster by exploiting the structure of D. For simplicity, assume that all the weights are different. Let $R \searrow S \searrow T$ be three consecutive rectangles in Q^*, then we can extract information about S. Consider the following cases.

If $R_x < S_x$ and $S_x < T_x$, then (i) S is the heaviest rectangle to the right of R with corner $B(S)$.
If $R_x < S_x$ and $S_y > T_y$, then (ii) S is the heaviest rectangle with corner $A(S)$.
If $R_y > S_y$ and $S_x < T_x$, then (iii) S is the heaviest rectangle with corner $B(S)$.
If $R_y > S_y$ and $S_y > T_y$, then (iv) S is the heaviest rectangle below R with corner $A(S)$.

Note that for each Property (i)–(iv), the rectangle S depends on a single parameter associated to S and on at most one parameter associated to R. More precisely, the rectangle S with Properties (i), (ii), (iii) and (iv) is completely determined by the parameters $\{B(R)_x, B(S)\}$, $\{A(S)\}$, $\{B(S)\}$ and $\{A(R)_y, A(S)\}$, respectively. This motivates the following recursion. For $R \in \mathcal{R}$, let $V(R)$ be the maximum weight of a path in D starting with R. For $a \in A$ (resp. $b \in B$), let (resp. $V_{\rightarrow}(b)$) be the maximum weight of a path using only rectangles below a (resp. to the right of b). We have

$$V(R) = \max\{V_{\downarrow}(A(R)), V_{\rightarrow}(B(R))\} + w_R,$$
$$V_{\downarrow}(a) = \max\{V(S)\colon S \text{ rectangle below } a \text{ satisfying (iii) or (iv)}\},$$
$$V_{\rightarrow}(b) = \max\{V(S)\colon S \text{ rectangle to the right of } b \text{ satisfying (i) or (ii)}\}.$$

Given $\{V(R)\}_{R\in\mathcal{R}}$, we can easily compute the optimal path Q^*. Note that evaluating $V_{\downarrow}(a)$ (or $V_{\rightarrow}(b)$) requires to compute the maximum of $V(S)$ over a family of $O(n)$ rectangles S. If this family can be computed in $O(n)$ time, then the recursive formula for V,$V_{\downarrow}$ and $V_{\rightarrow}$ can be completely evaluated in $O(n^2)$ time. We achieve this by precomputing all possible arising families in $O(n^2)$ time. We illustrate this only for rectangles having Property (i). Suppose $B(S) = b$. Traversing the points $b' \in B$ from right to left, we can find the heaviest such rectangle S to the right of b', for all $b' \in B$, in $O(n)$ time. Iterating this for every b, we compute all the families of rectangles arising from Property (i) in $O(n^2)$ time.

If there are repeated weights, we break ties in Properties (i) and (iii) by choosing the rectangle S of smallest width and we break ties in Properties (ii) and (iv) by choosing the rectangle S of smallest height.

Theorem 8. *The maximum weight cross-free matching of a bipartite permutation graph can be computed in $O(n^2)$ time.*

Using these ideas on the weighted 0-1 case, we can compute an optimum solution in $O(n)$ time, under certain assumption about the description of the input. We defer the description of the algorithm to the full version of the paper.

By the discussion in Sect. 5, cross-free matchings of convex graphs correspond to independent sets of a certain system of point-interval pairs. Lubiw [23] gives a polynomial time algorithm for the maximum weight of a system of point-interval pairs. It is straightforward to implement her algorithm in $O(n^3)$ time.

Corollary 2. *The maximum weight cross-free matching of a convex graph can be computed in $O(n^3)$ time.*

7 Summary of Results

We include a table with the current best running times for the studied problem on a graph $G = (A \cup B, \mathcal{R})$, with $n = |A \cup B|$. The new results, in bold, include the ones obtained via their relation to other known problems, but never considered in the literature of the problems we have studied.

	Bip. Perm.	Biconvex	Convex	2-dorg
Max. cross-free matching	$O(n)$ [3,12]	$O(n^2)$ [3]	$O(n^9)$ [11]	-
(Jump number) **(new)**	-	-	$\boldsymbol{O(n^2)}$[a]	$\boldsymbol{\widetilde{O}(n^{2.5})}, \boldsymbol{\widetilde{O}(n^{\omega})}$[c]
Min. biclique-cover	$O(n)$ [3,12]	-	-	-
(new)	-	$\boldsymbol{O(n^2)}$[a]	$\boldsymbol{O(n^2)}$[a]	$\boldsymbol{\widetilde{O}(n^{2.5})}, \boldsymbol{\widetilde{O}(n^{\omega})}$[c]
Max. wt. cross-free matching	$O(\lvert\mathcal{R}\rvert^2)$ [3]	-	-	-
(new)	$\boldsymbol{O(n^2)}$	$\boldsymbol{O(n^3)}$[b]	$\boldsymbol{O(n^3)}$[b]	**NP-hard**

[a] Using Franzblau and Kleitman's result [18] or Knuth's [21] implementation.
[b] Using Lubiw's result [23].
[c] In the full version of this paper.

References

1. Amilhastre, J., Janssen, P., Vilarem, M.C.: Computing a minimum biclique cover is polynomial for bipartite domino-free graphs. In: Proceedings of the Eight Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1997, pp. 36–42 (1997)
2. Benczúr, A.A.: Pushdown-reduce: An algorithm for connectivity augmentation and poset covering problems. Discrete Appl. Math. 129(2-3), 233–262 (2003)
3. Brandstädt, A.: The jump number problem for biconvex graphs and rectangle covers of rectangular regions. In: Csirik, J., Demetrovics, J., Gécseg, F. (eds.) FCT 1989. LNCS, vol. 380, pp. 68–77. Springer, Heidelberg (1989)
4. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph classes: A survey. SIAM, Philadelphia (1999)
5. Ceroi, S.: Ordres et géométrie plane: Application au nombre de sauts. Ph.D. thesis, Université Montpellier II (2000)
6. Ceroi, S.: A weighted version of the jump number problem on two-dimensional orders is NP-complete. Order 20(1), 1–11 (2003)
7. Chaiken, S., Kleitman, D.J., Saks, M., Shearer, J.: Covering regions by rectangles. SIAM J. Algebra Discr. 2(4), 394–410 (1981)
8. Chaty, G., Chein, M.: Ordered matchings and matchings without alternating cycles in bipartite graphs. Utilitas Math. 16, 183–187 (1979)
9. Cohen, B., Skiena, S.: Optimizing combinatorial library construction via split synthesis. In: Proceedings of the Third Annual International Conference on Research in Computational Molecular Biology, RECOMB 1999, pp. 124–133 (1999)
10. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to algorithms, 3rd edn. MIT Press, Cambridge (2009)
11. Dahlhaus, E.: The computation of the jump number of convex graphs. In: Bouchitté, V., Morvan, M. (eds.) ORDAL 1994. LNCS, vol. 831, pp. 176–185. Springer, Heidelberg (1994)
12. Fauck, H.: Covering polygons with rectangles via edge coverings of bipartite permutation graphs. J. Inform. Process. Cybernet. 27(8), 391–409 (1991)
13. Ford, L., Fulkerson, D.: Flows in networks. Princeton University Press, Princeton (2010)
14. Fowler, R.J., Paterson, M., Tanimoto, S.L.: Optimal packing and covering in the plane are NP-complete. Inf. Process. Lett. 12(3), 133–137 (1981)
15. Frank, A.: Finding minimum generators of path systems. J. Comb. Theory, Ser. B 75(2), 237–244 (1999)

16. Frank, A., Jordán, T.: Minimal edge-coverings of pairs of sets. J. Comb. Theory, Ser. B 65(1), 73–110 (1995)
17. Frank, A., Végh, L.A.: An algorithm to increase the node-connectivity of a digraph by one. Discrete Optimization 5(4), 677–684 (2008)
18. Franzblau, D.S., Kleitman, D.J.: An algorithm for covering polygons with rectangles. Inform. and Control 63(3), 164–189 (1984)
19. Györi, E.: A minimax theorem on intervals. J. Comb. Theory, Ser. B 37(1), 1–9 (1984)
20. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. 2(4), 225–231 (1973)
21. Knuth, D.E.: Irredundant intervals. ACM J. Exp. Algorithmics 1 (1996)
22. Kushilevitz, E., Nisan, N.: Communication complexity. Cambridge University Press, New York (1997)
23. Lubiw, A.: A weighted min-max relation for intervals. J. Comb. Theory, Ser. B 53(2), 151–172 (1991)
24. Mucha, M., Sankowski, P.: Maximum matchings via gaussian elimination. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2004, pp. 248–255 (2004)
25. Müller, H.: Alternating cycle-free matchings. Order 7, 11–21 (1990)
26. Müller, H.: On edge perfectness and classes of bipartite graphs. Discrete Mathematics 149(1-3), 159–187 (1996)
27. Nau, D.S., Markowsky, G., Woodbury, M.A., Amos, D.B.: A mathematical analysis of human leukocyte antigen serology. Math. Biosci. 40(3-4), 243–270 (1978)
28. Orlin, J.: Contentment in graph theory: Covering graphs with cliques. Indagationes Mathematicae (Proceedings) 80(5), 406–424 (1977)
29. Otachi, Y., Okamoto, Y., Yamazaki, K.: Relationships between the class of unit grid intersection graphs and other classes of bipartite graphs. Discrete Applied Mathematics 155(17), 2383–2390 (2007)
30. Pulleyblank, W.R.: Alternating cycle free matchings. Tech. Rep. CORR 82-18, University of Waterloo - Dept. of Combinatorics and Optimization (1982)
31. Schrijver, A.: Combinatorial Optimization - Polyhedra and Efficiency. Springer, Berlin (2003)
32. Shrestha, A.M., Tayu, S., Ueno, S.: On two-directional orthogonal ray graphs. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems, ISCAS 2010, pp. 1807–1810 (2010)
33. Steiner, G., Stewart, L.K.: A linear time algorithm to find the jump number of 2-dimensional bipartite partial orders. Order 3, 359–367 (1987)
34. Végh, L.A.: Connectivity Augmentation Algorithms. Ph.D. thesis, Eötvös Loránd University (2010)

Optimal Matching Forests and Valuated Delta-Matroids

Kenjiro Takazawa

Research Institute for Mathematical Sciences,
Kyoto University, Kyoto 606-8502, Japan
takazawa@kurims.kyoto-u.ac.jp

Abstract. The matching forest problem in mixed graphs is a common generalization of the matching problem in undirected graphs and the branching problem in directed graphs. Giles presented an $O(n^2m)$-time algorithm for finding a maximum-weight matching forest, where n is the number of vertices and m is that of edges, and a linear system describing the matching forest polytope. Later, Schrijver proved total dual integrality of the linear system. In the present paper, we reveal another nice property of matching forests: the degree sequences of the matching forests in any mixed graph form a delta-matroid and the weighted matching forests induce a valuated delta-matroid. We remark that the delta-matroid is not necessarily even, and the valuated delta-matroid induced by weighted matching forests slightly generalizes the well-known notion of Dress and Wenzel's valuated delta-matroids. By focusing on the delta-matroid structure and reviewing Giles' algorithm, we design a simpler $O(n^2m)$-time algorithm for the weighted matching forest problem. We also present a faster $O(n^3)$-time algorithm by using Gabow's method for the weighted matching problem.

Keywords: Matching, Branching, Matching Forest, Delta-Matroid, Valuated Delta-Matroid, Primal-Dual Algorithm.

1 Introduction

The concept of *matching forests* in mixed graphs was introduced by Giles [14–16] as a common generalization of matchings in undirected graphs and branchings in directed graphs. Let $G = (V, E, A)$ be a mixed graph with vertex set V, undirected edge set E and directed edge set A. Let n and m denote $|V|$ and $|E \cup A|$, respectively. For $x \in \mathbf{R}^{E \cup A}$ and $F \subseteq E \cup A$, let $x(F) := \sum_{e \in F} x(e)$.

We denote a directed edge $a \in A$ from $u \in V$ to $v \in V$ by uv. A directed edge is often called an *arc*. For an arc $a = uv$, the terminal vertex v is called the *head* of a and denoted by $\partial^- a$, and the initial vertex u is called the *tail* of a and denoted by $\partial^+ a$. For a vertex $v \in V$, the set of arcs whose head (resp., tail) is v is denoted by $\delta^- v$ (resp., $\delta^+ v$). For $B \subseteq A$, let $\partial^- B = \bigcup_{a \in B} \partial^- a$. A vertex in $\partial^- B$ is said to be *covered* by B. An arc subset $B \subseteq A$ is a *branching* if the underlying edge set of B is a forest and each vertex is the head of at most one

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 404–416, 2011.

edge in B. For a branching B, a vertex not covered by B is called a *root* of B, and the set of the roots of B is denoted by $R(B)$, i.e., $R(B) = V \setminus \partial^- B$.

An undirected edge $e \in E$ connecting $u, v \in V$ is denoted by (u, v). We often abbreviate (u, v) as uv, where it obvious that it is undirected. For $e = uv \in E$, both u and v are called as the *head* of e, and the set of heads of e is denoted by ∂e, i.e., $\partial e = \{u, v\}$. For a vertex v, the set of edges incident to v is denoted by δv. For $F \subseteq E$, let $\partial F = \bigcup_{e \in F} \partial e$. A vertex in ∂F is said to be *covered* by F. An undirected edge subset $M \subseteq E$ is a *matching* if each vertex is the head of at most one edge in M. A vertex not covered by M is called a *root* of M and the set of the roots of M is denoted by $R(M)$, i.e., $R(M) = V \setminus \partial M$.

An edge set $F \subseteq E \cup A$ is a *matching forest* if the underlying edge set of F is a forest and each vertex is the head of at most one edge in F. Equivalently, an edge set $F = B \cup M$, where $B \subseteq A$ and $M \subseteq E$, is a matching forest if B is a branching and M is a matching with $\partial M \subseteq R(B)$. A vertex is a *root* of a matching forest F if it is not covered by F, and the set of the roots of F is denoted by $R(F)$. Observe that $R(F) = R(M) \cap R(B)$ and $V = R(M) \cup R(B)$.

Matching forests inherit the tractability of branchings and matchings. Let $w \in \mathbf{R}^{E \cup A}$ be a weight vector on the edge set of a mixed graph $G = (V, E, A)$. We consider the *weighted matching forest problem*, the objective of which is to find a matching forest F maximizing $w(F)$. For this problem, Giles [15] designed a primal-dual algorithm running in $O(n^2 m)$ time, which provided a constructive proof for integrality of a linear system describing the matching forest polytope. Later, Schrijver [21] proved that Giles' linear system is totally dual integral. These results commonly extend the polynomial-time solvability and the total dual integrality results for the weighted branchings and matchings [4, 7, 9].

Topics related to matching forests include the following. Using the notion of matching forests, Keijsper [17] gave a common extension of Vizing's theorem [23, 24] on covering undirected graphs by matchings and Frank's theorem [11] on covering directed graphs by branchings. Another aspect of matching forests is that they can be represented as *linear matroid matching* (see [22]). From this viewpoint, however, we do not fully understand the tractability of matching forests, since the weighted linear matroid matching problem is unsolved while the unweighted problem is solved [18].

In the present paper, we reveal a relation between matching forests and *delta-matroids* [1, 3, 5] to offer a new perspective on weighted matching forests which explains their tractability. For a finite set V and $\mathcal{F} \subseteq 2^V$, the pair $(V, \mathcal{F})$ is a delta-matroid if it satisfies the following exchange property:

(DM) $\forall S_1, S_2 \in \mathcal{F}$, $\forall s \in S_1 \triangle S_2$, $\exists t \in S_1 \triangle S_2$, $S_1 \triangle \{s, t\} \in \mathcal{F}$.

Here, $\triangle$ denotes the symmetric difference, i.e., $S_1 \triangle S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$.

A typical example of a delta-matroid is a *matching delta-matroid*. For an undirected graph $G = (V, E)$, let $\mathcal{F}_M = \{\partial M \mid M \text{ is a matching in } G\}$. Then, $(V, \mathcal{F}_M)$ is a delta-matroid [2, 3]. Branchings in a directed graph also induce a delta-matroid, which we call a *branching delta-matroid*. For a directed graph $G = (V, A)$, let $\mathcal{F}_B = \{R(B) \mid B \text{ is a branching in } G\}$. Then, it is not difficult to see that $\mathcal{F}_B$ is a delta-matroid (see § 2.1).

A delta-matroid $(V, \mathcal{F})$ is called *even* if $|S_1| - |S_2|$ is even for any $S_1, S_2 \in \mathcal{F}$. Note that a matching delta-matroid is an even delta-matroid, whereas a branching delta-matroid is not. Even delta-matroids are characterized by the following simultaneous exchange property [25]:

(EDM) $\forall S_1, S_2 \in \mathcal{F}$, $\forall s \in S_1 \triangle S_2$, $\exists t \in (S_1 \triangle S_2) \setminus \{s\}$, $S_1 \triangle \{s,t\} \in \mathcal{F}$ and $S_2 \triangle \{s,t\} \in \mathcal{F}$.

The concept of *valuated delta-matroids* [6, 26] is a quantitative generalization of even delta-matroids. A function $f : 2^V \to \mathbf{R} \cup \{-\infty\}$ is a valuated delta-matroid if $\mathrm{dom} f \neq \emptyset$ and

(V-EDM) $\forall S_1, S_2 \in \mathrm{dom} f$, $\forall s \in S_1 \triangle S_2$, $\exists t \in (S_1 \triangle S_2) \setminus \{s\}$, $f(S_1 \triangle \{s,t\}) + f(S_2 \triangle \{s,t\}) \geq f(S_1) + f(S_2)$.

Here, $\mathrm{dom} f := \{S \mid S \subseteq V,\ f(S) \neq -\infty\}$. Note that $(V, \mathrm{dom} f)$ is an even-delta matroid. We remark here that weighted matchings in a weighted undirected graph induce a valuated delta-matroid f_M with $\mathrm{dom} f_M = \mathcal{F}_M$ (see § 2.1).

In this paper, we consider delta-matroids commonly extending matching delta-matroids and branching delta-matroids, and also a valuation on those delta-matroids. For this purpose, we introduce a new class of delta-matroids which properly includes even delta-matroids. We call $(V, \mathcal{F})$ a *simultaneous delta-matroid* if it satisfies the following weaker simultaneous exchange property:

(SDM) $\forall S_1, S_2 \in \mathcal{F}, \forall s \in S_1 \triangle S_2, \exists t \in S_1 \triangle S_2$, $S_1 \triangle \{s,t\} \in \mathcal{F}$ and $S_2 \triangle \{s,t\} \in \mathcal{F}$.

Note that every even delta-matroid is a simultaneous delta-matroid. Also, a branching matroid is a simultaneous delta-matroid (see § 2.1).

The first main result in this paper is that matching forests also induce a simultaneous delta-matroid. For a mixed graph $G = (V, E, A)$, let $\mathcal{F}_{MF} = \{R(F) \mid F \text{ is a matching forest in } G\}$.

Theorem 1. *For a mixed graph $G = (V, E, A)$, $(V, \mathcal{F}_{MF})$ is a simultaneous delta-matroid.*

Furthermore, we generalize the notion of valuated delta-matroids in order to deal with a quantitative extension of Theorem 1. That is, we define valuated delta-matroids on simultaneous delta-matroids, which slightly generalize valuated delta-matroids defined above. We call a function $f : 2^V \to \mathbf{R} \cup \{-\infty\}$ a *valuated delta-matroid* if $\mathrm{dom} f \neq \emptyset$ and

(V-SDM) $\forall S_1, S_2 \in \mathrm{dom} f$, $\forall s \in S_1 \triangle S_2$, $\exists t \in S_1 \triangle S_2$, $f(S_1 \triangle \{s,t\}) + f(S_2 \triangle \{s,t\}) \geq f(S_1) + f(S_2)$.

Note that $(V, \mathrm{dom} f)$ is a simultaneous delta-matroid.

For a weighted mixed graph (G, w) with $G = (V, E, A)$ and $w \in \mathbf{R}^{E \cup A}$, define a function $f_{MF} : 2^V \to \mathbf{R} \cup \{-\infty\}$ by

$$f_{MF}(S) = \begin{cases} \max\{w(F) \mid F \text{ is a matching forest with } R(F) = S\} & (S \in \mathcal{F}_{MF}), \\ -\infty & (\text{otherwise}). \end{cases}$$

We prove that f_{MF} satisfies (V-SDM).

Theorem 2. *For a weighted mixed graph (G, w), f_{MF} is a valuated delta-matroid.*

Proofs for Theorems 1 and 2 will be given in § 2.2. We remark that the relation between valuated delta-matroids in the sense of [6] and those in our sense is similar to that between M-concave functions and $M^\natural$-concave functions [20].

The next contribution of this paper is new algorithms for the weighted matching forest problem: we design a simpler algorithm and a faster algorithm than Giles' algorithm [15]. In § 3, we present a simple $O(n^2m)$ algorithm which focuses on the delta-matroid structure. We also present a faster $O(n^3)$ algorithm in § 4 by using the technique of Gabow [13] for the weighted matching problem.

2 Delta-Matroids and Matching Forests

2.1 Matching Delta-Matroids and Branching Delta-Matroids

In this subsection, we describe basic facts on delta-matroids, including their relations to matchings and branchings. The *dual* of a delta-matroid $(V, \mathcal{F})$ is a delta-matroid $(V, \bar{\mathcal{F}})$, defined by $\bar{\mathcal{F}} = \{V \setminus S \mid S \in \mathcal{F}\}$. The *union* of two delta-matroids $(V, \mathcal{F}_1)$ and $(V, \mathcal{F}_2)$ is a pair $(V, \mathcal{F}_1 \vee \mathcal{F}_2)$ defined by $\mathcal{F}_1 \vee \mathcal{F}_2 = \{S_1 \cup S_2 \mid S_1 \in \mathcal{F}_1, S_2 \in \mathcal{F}_2, S_1 \cap S_2 = \emptyset\}$, which is a delta-matroid [2].

Let (G, w) be a weighted undirected graph with $G = (V, E)$ and $w \in \mathbf{R}^E$. As stated in § 1, the pair $(V, \mathcal{F}_M)$, where $\mathcal{F}_M = \{\partial M \mid M \text{ is a matching in } G\}$, is an even delta-matroid, which we call the matching delta-matroid of G. Moreover, a function $f_M : 2^V \to \mathbf{R} \cup \{-\infty\}$ defined below is a valuated delta-matroid [19]:

$$f_M(S) = \begin{cases} \max\{w(M) \mid M \text{ is a matching with } \partial M = S\} & (S \in \mathcal{F}_M), \\ -\infty & (\text{otherwise}). \end{cases}$$

We now present a relation between branchings and delta-matroids. Let (G, w) be a weighted directed graph with $G = (V, A)$ and $w \in \mathbf{R}^A$. Recall that $\mathcal{F}_B = \{R(B) \mid B \text{ is a branching in } G\}$. It is verified that $(V, \mathcal{F}_B)$ is a delta-matroid as follows. For a directed graph G, a strong component is called a *source component* if it has no arc entering from other strong components. The vertex set and arc set of a strong component K are denoted by VK and AK, respectively. Let $K_1, \ldots, K_l$ be all source components in G. Then, we have that $\mathcal{F}_B = \{S \mid S \subseteq V, |S \cap VK_i| \geq 1 \text{ for } i = 1, \ldots, l\}$. Thus, it follows that $(V, \mathcal{F}_B)$ is a generalized matroid [12]. Moreover, it also follows that $(V, \mathcal{F}_B)$ satisfies (SDM). We call $(V, \mathcal{F}_B)$ as the branching delta-matroid of G.

Theorem 3. *For a directed graph G, $(V, \mathcal{F}_B)$ is a simultaneous delta-matroid.*

This fact extends to weighted branchings. Define $f_B : 2^V \to \mathbf{R} \cup \{-\infty\}$ by

$$f_B(S) = \begin{cases} \max\{w(B) \mid B \text{ is a branching with } R(B) = S\} & (S \in \mathcal{F}_B), \\ -\infty & (\text{otherwise}). \end{cases}$$

Then, f_B is a valuated delta-matroid, which immediately follows from arguments in Schrijver [21, Theorem 1].

Theorem 4. *For a weighted directed graph (G, w), f_B is a valuated delta-matroid.*

2.2 Delta-Matroids and Matching Forests

In this subsection, we prove Theorems 1 and 2. We begin with a simple proof showing that $(V, \mathcal{F}_{MF})$ is a delta-matroid for a mixed graph (V, E, A). Let $(V, \mathcal{F}_M)$ be the matching delta-matroid of (V, E) and $(V, \mathcal{F}_B)$ the branching delta-matroid of (V, A). Then, it follows from the definition of matching forests that $\mathcal{F}_{MF}$ is the dual of $\mathcal{F}_M \vee \bar{\mathcal{F}}_B$, and thus $(V, \mathcal{F}_{MF})$ is a delta-matroid.

We now prove Theorem 1, which is a stronger statement. First, Schrijver [21] proved the following exchange property of branchings.

Lemma 1 (Schrijver [21]). *Let $G = (V, A)$ be a directed graph, and B_1 and B_2 be branchings partitioning A. Let R_1 and R_2 be vertex sets with $R_1 \cup R_2 = R(B_1) \cup R(B_2)$ and $R_1 \cap R_2 = R(B_1) \cap R(B_2)$. Then A can be split into branchings B_1' and B_2' with $R(B_i') = R_i$ for $i = 1, 2$ if and only if each source component K in G satisfies that $|K \cap R_i| \geq 1$ for $i = 1, 2$.*

By using Lemma 1, Schrijver proved an exchange property of matching forests [21, Theorem 2]. Here, we show another exchange property of matching forests. The proof below is quite similar to the proof for Theorem 2 in [21]. For completeness, however, we describe a full proof.

Lemma 2. *Let $G = (V, E, A)$ be a mixed graph, F_1 and F_2 be matching forests partitioning $E \cup A$, and $s \in R(F_2) \setminus R(F_1)$. Then, there exist matching forests F_1' and F_2' which partition $E \cup A$ and satisfy one of the following:*

1. *$R(F_1') = R(F_1) \cup \{s\}$ and $R(F_2') = R(F_2) \setminus \{s\}$,*
2. *$R(F_1') = R(F_1) \cup \{s, t\}$ and $R(F_2') = R(F_2) \setminus \{s, t\}$ for some $t \in R(F_2) \setminus (R(F_1) \cup \{s\})$,*
3. *$R(F_1') = (R(F_1) \cup \{s\}) \setminus \{t\}$ and $R(F_2') = (R(F_2) \setminus \{s\}) \cup \{t\}$ for some $t \in R(F_1) \setminus R(F_2)$.*

Proof. Let $M_i = F_i \cap E$ and $B_i = F_i \cap A$ for $i = 1, 2$. Denote the family of the source components in (V, A) by $\mathcal{K}$. If $v \in R(B_1) \cap R(B_2)$ for $v \in V$, then we have $\{v\} \in \mathcal{K}$. Thus, for a source component $K \in \mathcal{K}$ with $|K| \geq 2$, $K \cap R(B_1)$ and $K \cap R(B_2)$ are not empty and disjoint with each other. For each $K \in \mathcal{K}$ with $|K| \geq 2$, choose a pair e_K of vertices, one of which is in $K \cap R(B_1)$ and the other in $K \cap R(B_2)$. Denote $N = \{e_K \mid K \in \mathcal{K}\}$. Note that N is a matching.

Construct an undirected graph $H = (V, M_1 \cup M_2 \cup N)$. We have that H is a disjoint collection of paths and cycles. For, an endpoint u of an edge $e_K \in N$ satisfies that either $u \in \partial^- B_1$ or $u \in \partial^- B_2$, and thus u is not covered by both of M_1 and M_2. Moreover, s is an endpoint of a path P in H. For, since $s \in R(F_2)$, we have that s is not covered by M_2. If s is covered by M_1, then $s \in R(B_1)$, and thus $s \in R(B_1) \cap R(B_2)$. This implies that s is not covered by N.

Denote the set of vertices on P by VP, the set of edges in $M_1 \cup M_2$ on P by EP, and let $M_1' := M_1 \triangle EP$ and $M_2' := M_2 \triangle EP$. Then, both M_1' and M_2' are matchings and $R(M_1') = (R(M_1) \setminus VP) \cup (R(M_2) \cap VP)$ and $R(M_2') = (R(M_2) \setminus VP) \cup (R(M_1) \cap VP)$. Now, by Lemma 1, there exist disjoint branchings B_1' and

B'_2 such that $R(B'_1) = (R(B_1) \setminus VP) \cup (R(B_2) \cap VP)$ and $R(B'_2) = (R(B_2) \setminus VP) \cup (R(B_1) \cap VP)$. (Note that $|K \cap R(B'_i)| \geq 1$ for $i = 1, 2$ for every source component K.)

Since $R(B_i) \cup R(M_i) = V$, we have that $F'_i = B'_i \cup M'_i$ is a matching forest for $i = 1, 2$, and $R(F'_1) = (R(F_1) \setminus VP) \cup (R(F_2) \cap VP)$ and $R(F'_2) = (R(F_2) \setminus VP) \cup (R(F_1) \cap VP)$. If $VP = \{s\}$, then Assertion 1 applies. Otherwise, denote the other endpoint of P by t. If $t \in V \setminus (R(F_1) \triangle R(F_2))$, then Assertion 1 applies. If $t \in R(F_2) \setminus R(F_1)$, then Assertion 2 applies. If $t \in R(F_1) \setminus R(F_2)$, then Assertion 3 applies.

Theorem 1 is obvious from Lemma 2. Furthermore, Theorem 2 also follows from Lemma 2.

Proof for Theorem 2. Let $S_1, S_2 \in \mathrm{dom} f$ and $s \in S_1 \triangle S_2$. For $i = 1, 2$, let F_i be a matching forest such that $R(F_i) = S_i$ and $w(F_i) = f_{MF}(S_i)$. Without loss of generality, assume $s \in R(F_2) \setminus R(F_1)$. By applying Lemma 2 to the mixed graph consisting of the edges in F_1 and F_2, we obtain matching forests F'_1 and F'_2 such that $w(F'_1) + w(F'_2) = w(F_1) + w(F_2)$ and satisfying one of Assertions 1–3. Now the statement follows from $w(F'_1) \leq f_{MF}(R(F'_1))$ and $w(F'_2) \leq f_{MF}(R(F'_2))$.

3 A Simpler Algorithm

Let (G, w) be a weighted mixed graph with $G = (V, E, A)$ and $w \in \mathbf{R}^{E \cup A}$. In this section, we describe a primal-dual algorithm for finding a matching forest F maximizing $w(F)$. It is a slight modification of Giles' algorithm [15]. The main difference results from focusing on branching delta-matroids (Theorem 3).

3.1 Linear Programming Formulation

For a subpartition $\mathcal{L}$ of V, let $\cup\mathcal{L}$ denote the union of the sets in $\mathcal{L}$ and let

$$\gamma(\mathcal{L}) := \{e \mid e \in E,\ e \text{ is contained in } \cup\mathcal{L}\} \cup \{a \mid a \in A,\ a \text{ is contained in some set in } \mathcal{L}\}.$$

Let Λ denote the collection of subpartition $\mathcal{L}$ of V with $|\mathcal{L}|$ odd. The following is a linear programming relaxation of an integer program describing the weighted matching forest problem:

$$\begin{aligned}
\text{(P)}\quad \text{maximize} \quad & \sum_{e \in E \cup A} w(e)x(e) & & \\
\text{subject to} \quad & x(\delta^{\mathrm{head}}(v)) \leq 1 & (v \in V), & \quad (1)\\
& x(\gamma(\mathcal{L})) \leq \lfloor |\cup\mathcal{L}| - |\mathcal{L}|/2 \rfloor & (\mathcal{L} \in \Lambda), & \quad (2)\\
& x(e) \geq 0 & (e \in E \cup A). & \quad (3)
\end{aligned}$$

Here, $\delta^{\mathrm{head}}(v) \subseteq E \cup A$ denotes the set of edges which have v as a head, i.e., $\delta^{\mathrm{head}}(v) = \delta v \cup \delta^- v$. Note that the above linear system is a common extension of those describing the weighted matching problem [7] and the weighted branching problem [9]. Giles [15] proved the integrality of the system (1)–(3).

Furthermore, Schrijver [21] proved the total dual integrality of the system (1)–(3), which commonly extends the total dual integrality of matching constraints [4] and branching constraints. That is, Schrijver proved that the following dual problem of (P) has an integer optimal solution if w is integer:

$$\begin{aligned}
\text{(D)}\quad \text{minimize}\quad & \sum_{v\in V} y(v) + \sum_{\mathcal{L}\in\Lambda} z(\mathcal{L})\lfloor |\cup\mathcal{L}| - |\mathcal{L}|/2\rfloor & & \\
\text{subject to}\quad & y(u)+y(v)+\sum_{\mathcal{L}:\, e\in\gamma(\mathcal{L})} z(\mathcal{L}) \geq w(e) & (e=uv\in E), & \quad (4)\\
& y(v)+\sum_{\mathcal{L}:\, a\in\gamma(\mathcal{L})} z(\mathcal{L}) \geq w(a) & (a=uv\in A), & \quad (5)\\
& y(v)\geq 0 & (v\in V), & \quad (6)\\
& z(\mathcal{L})\geq 0 & (\mathcal{L}\in\Lambda). & \quad (7)
\end{aligned}$$

Define the reduced weight $w' \in \mathbf{R}^{E\cup A}$ by $w'(e) = y(u) + y(v) + \sum_{\mathcal{L}:\, e\in\gamma(\mathcal{L})} z(\mathcal{L}) - w(e)$ for $e = uv \in E$ and $w'(a) = y(v) + \sum_{\mathcal{L}:\, a\in\gamma(\mathcal{L})} z(\mathcal{L}) - w(a)$ for $a = uv \in A$. Below are the complementary slackness conditions of (P) and (D).

$$\begin{aligned}
x(e) > 0 &\implies w'(e) = 0 && (e\in E\cup A), && (8)\\
x(\delta^{\mathrm{head}} v) < 1 &\implies y(v) = 0 && (v\in V), && (9)\\
z(\mathcal{L}) > 0 &\implies x(\gamma(\mathcal{L})) = \lfloor |\cup\mathcal{L}| - |\mathcal{L}|/2\rfloor && (\mathcal{L}\in\Lambda). && (10)
\end{aligned}$$

3.2 Algorithm Description

In the algorithm, we keep a matching forest F, which corresponds to an integer feasible solution x of (P), and a dual feasible solution (y, z). We maintain that x and (y, z) satisfy (8) and (10). The algorithm terminates when (9) is satisfied.

Similarly to the classical weighted matching and branching algorithms, we execute shrinking of subgraphs repeatedly. We keep two laminar families Δ and Υ of subsets of V, the former of which results from shrinking a strong component in the directed graph and the latter from shrinking an undirected odd cycle.

We use the following notations to describe the algorithm.

- For a cycle or a path Q in an undirected graph (V, E), let VQ and EQ denote the vertex set and edge set of Q, respectively. We often abbreviate EQ as Q.
- $\Omega' := \Delta\cup\Upsilon$, $\Omega := \Omega'\cup\{\{v\} \mid v\in V\}$.
- For each $U\in\Omega'$, let $G_U = (V_U, E_U, A_U)$ denote the mixed graph obtained from the subgraph induced by U by contracting all maximal proper subsets of U belonging to Δ. Also, let $\hat{G} = (\hat{V}, \hat{E}, \hat{A})$ denote the mixed graph obtained from G by contracting all maximal sets in Ω'. We denote a vertex in a shrunk graph by the set of vertices in V which are shrunk into the vertex. Also, we often identify a vertex U in a shrunk graph and the singleton $\{U\}$.

– For $G = (V, E, A)$ and a dual feasible solution (y, z), the *equality subgraph* $G^\circ = (V, E^\circ, A^\circ)$ is a subgraph of G defined by $E^\circ = \{e \mid e \in E, w'(e) = 0\}$ and $A^\circ = \{a \mid a \in A, w'(a) = 0\}$. We denote the branching delta-matroid in $(\hat{V}, \hat{A}^\circ)$ by $(\hat{V}, \hat{\mathcal{F}}_B^\circ)$, i.e., $\hat{\mathcal{F}}_B^\circ = \{R(\hat{B}) \mid \hat{B} \subseteq \hat{A}^\circ \text{ is a branching in } \hat{G}^\circ\}$.

The outline of the algorithm is as follows.

– We maintain a matching forest $\hat{F} = \hat{M} \cup \hat{B}$ in $\hat{G}^\circ$, where $\hat{M} \subseteq \hat{E}^\circ$ and $\hat{B} \subseteq \hat{A}^\circ$, in order to maintain (8).
– In contracting a vertex set $U \subseteq V$, we associate a partition $\mathcal{L}_U$ of U such that $x(\gamma(\mathcal{L}_U)) = \lfloor |\cup \mathcal{L}_U| - |\mathcal{L}_U|/2 \rfloor$. The vector z is restricted to subpartitions associated to the sets in Ω' in order to maintain (10).
– Similarly to Edmonds' matching algorithm [8], we construct an *alternating forest* H, which is a subgraph of $(\hat{V}, \hat{E}^\circ)$. The vertex set and edge set of H are denoted by $\hat{V}H$ and $\hat{E}H$, respectively. We often abbreviate $\hat{E}H$ as H. Each component of H is a tree and contains a unique *source vertex*. Intuitively, a source vertex is a vertex where (9) is not satisfied (see Step 2 below for precise definition). For $v \in \hat{V}H$, let P_v denote the path in H connecting a source vertex and v. The edges incident to a source vertex does not belong to $\hat{M}$, and edges in $\hat{M}$ and $\hat{E}^\circ \setminus \hat{M}$ appear alternately on each P_v. We label a vertex v as "even" (resp., "odd") if the length of P_v is even (resp., odd). Here, the length of a path is defined by the number of its edges. The set of vertices labelled as even (resp., odd) is denoted by $\mathrm{even}(H)$ (resp., $\mathrm{odd}(H)$). Also, let $\mathrm{free}(H) := \hat{V} \setminus (\mathrm{even}(H) \cup \mathrm{odd}(H))$.

We now present a full description of our algorithm.

Algorithm SIMPLE

Step 1. Set $F := \emptyset$, $y(v) := \max\{\{w(e)/2 \mid e \in E\}, \{w(a) \mid a \in A\}\}$ for every $v \in V$, $\Delta := \emptyset$ and $\Upsilon := \emptyset$.

Step 2. Construct the equality subgraph $\hat{G}^\circ = (\hat{V}, \hat{E}^\circ, \hat{A}^\circ)$. Define the set of source vertices $\hat{S} = \{U \mid U \in \hat{V}, y(v) > 0 \text{ and } x(\delta^{\mathrm{head}}(v)) = 0 \text{ for some } v \in U\}$. If $\hat{S} = \emptyset$, deshrink every sets in Ω' and return F. Otherwise, let H be $(\hat{S}, \emptyset)$, label the vertices in $\hat{S}$ as even, and then go to Step 3.

Step 3. If there exists an arc $a \in \hat{A}^\circ \setminus \hat{B}$ with $\partial^- a \in \mathrm{even}(H)$, then go to Step 4. Otherwise, go to Step 5.

Step 4. Let $v := \partial^- a$. If $R(\hat{B}) \setminus \{v\} \in \hat{\mathcal{F}}_B^\circ$, then go to Step 4.1. Otherwise, go to Step 4.2.

Step 4.1: Augmentation. Reset $\hat{F} := M' \cup B'$, where $M' := \hat{M} \triangle P_v$ and B' is a branching in $(\hat{V}, \hat{A}^\circ)$ with $R(B') = R(\hat{B}) \setminus \{v\}$. Delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω', and then go to Step 2.

Step 4.2. Let K be the source component containing v and let $X \subseteq V$ be the union of vertices in $\hat{V}K$. If there exists $e \in \hat{E}^\circ \setminus \hat{M}$ such that $\partial e \subseteq \hat{V}K$, then go to Step 4.2.1. Otherwise, go to Step 4.2.2.

Step 4.2.1: Augmentation. Let B_K be a branching in K with $R(B_K) = \partial e$. Reset $\hat{F} := M' \cup B'$, where $M' := (\hat{M} \triangle P_v) \cup \{e\}$ and $B' := (\hat{B} \setminus \hat{A}K) \cup B_K$, delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω', and then go to Step 2.

Step 4.2.2. Let $W_1, \ldots, W_l$ be the maximal proper subsets of X belonging to Υ. If, for some $i \in \{1, \ldots, l\}$, $\hat{A}K$ contains a pair of arcs $f^+ \in \delta^+ W_i$ and $f^- \in \delta^- W_i$ such that $\partial^+ f^+$ and $\partial^- f^-$ belong to distinct vertices in G_{W_i}, then go to Step 4.2.2.1. Otherwise, go to Step 4.2.2.2.

Step 4.2.2.1: Augmentation. Let B_K be a branching in K such that $R(B_K) = \{W_i\}$ and $f^+ \in B_K$. Reset $\hat{F} := M' \cup B'$, where $M' := \hat{M} \triangle P_v$ and $B' := (\hat{B} \setminus \hat{A}(K)) \cup B_K \cup \{f^-\}$. Then, delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω' and go to Step 2.

Step 4.2.2.2: Shrinking. For each $i = 1, \ldots, l$, let $v_{W_i} \in V_{W_i}$ denote the unique vertex in G_{W_i} to which arcs in $\hat{A}K$ are incident. Let $X' = (X \setminus (W_1 \cup \cdots \cup W_l)) \cup \{v_{W_i}\} \cup \cdots \cup \{v_{W_l}\}$ and add X' to Δ. Let $\mathcal{L}_{X'} = \{X'\}$ be the associated partition with X', set $z(\mathcal{L}_{X'}) := 0$, and then go to Step 3.

Step 5. Choose an edge $e \in \hat{E}^\circ \setminus \hat{E}H$ such that one of its head u is even. Denote the other head of e by v.

- If $v \in \mathrm{even}(H)$ and e connects different components in H, then go to Step 5.1.
- If $v \in \mathrm{even}(H)$ and u and v belong to the same component in H, then go to Step 5.2.
- If $v \in \mathrm{free}(H)$ and $v = \partial^- a$ for some $a \in \hat{B}$, then go to Step 5.3.
- If $v \in \mathrm{free}(H)$ and $v \in \partial e'$ for some $e' \in \hat{M}$, then go to Step 5.4.
- If v is a pseudo-vertex labelled as "saturated," then go to Step 5.5.

If no edge in $\hat{E}^\circ \setminus \hat{E}(H)$ satisfies the above conditions, then go to Step 6.

Step 5.1: Augmentation. Reset $\hat{F} := M' \cup \hat{B}$, where $M' := \hat{M} \triangle (P_u \cup P_v \cup \{e\})$, delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω', and then go to Step 2.

Step 5.2. Let C be the cycle in $H \cup \{e\}$ and let $U \subseteq V$ be the union of the vertices in $\hat{V}C$. Denote the maximal proper subsets of U belonging to Δ and Υ by $Y_1, \ldots, Y_k \in \Delta$ and $W_1 \ldots, W_l \in \Upsilon$, respectively. Let C_U be an odd cycle in G_U obtained by adding even number of edges from each E_{W_j} to C. For $i = 1, \ldots, k$, let $f_i^1, f_i^2 \in C_U$ denote the two edges incident to Y_i, and let $v_i^1, v_i^2 \in Y_i$ denote the vertices to which f_i^1 and f_i^2 are incident, respectively. If, for some Y_i, the two vertices v_i^1 and v_i^2 are distinct and $z(\mathcal{L}_{Y_i'}) = 0$ for the minimal subset $Y_i' \in \Delta$ of Y_i such that $\{v_i^1, v_i^2\} \subseteq Y_i'$, then go to Step 5.2.1. Otherwise, go to Step 5.2.2.

Step 5.2.1: Deshrinking and augmentation. Delete Y_i' from Δ and reset

$$\hat{M} := \begin{cases} (\hat{M} \triangle P_{Y_i}) \triangle C & (f_i^1, f_i^2 \in E \setminus M), \\ \hat{M} \triangle P_{Y_i}^* & (\text{otherwise}). \end{cases}$$

Here, $P_{Y_i}^*$ denotes the path in $H \cup \{e\}$ from $\hat{S}$ to Y_i consisting of odd number of edges. Delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω', and then go to Step 2.

Step 5.2.2: Shrinking. Without loss of generality, assume that v_i^1 and v_i^2 are identical for $i = 1, \ldots, j$, and distinct for $i = j+1, \ldots, k$. For $i = j+1, \ldots, k$, let $Y_i' \in \Delta$ be the minimal subset of Y_i such that $\{v_i^1, v_i^2\} \subseteq Y_i'$. Let $U' =$

$(U \setminus (Y_1 \cup \cdots \cup Y_k)) \cup \{v_{Y_1}, \ldots, v_{Y_j}\} \cup (Y'_{j+1} \cup \cdots \cup Y'_k)$. Add U' to Υ, and define a partition $\mathcal{L}_{U'}$ of U' by the collection of $\{v_{Y_1}\}, \ldots, \{v_{Y_j}\}, Y'_{j+1}, \ldots, Y'_k$ and singletons of the other vertices in U'. Set $z(\mathcal{L}_{U'}) := 0$ and then go to Step 3.

Step 5.3: Augmentation. Reset $\hat{F} := (\hat{M} \triangle P_v) \cup (\hat{B} \setminus \{a\})$, delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω', and then go to Step 2.

Step 5.4: Forest extension. Grow H by adding e and e'. Label v as odd and the other head of e' as even. Then, go to Step 3.

Step 5.5: Augmentation. Reset $\hat{F} := M' \cup \hat{B}$, where $M' := \hat{M} \triangle P_v$, and unlabel v. Delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ and then go to Step 2.

Step 6. Apply Dual_Update described below, delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω', and then go to Step 3.

Procedure Dual_Update. Define families of vertex subsets of V as follows:

$$\begin{aligned}
\Delta_+ &:= \{\text{maximal set in } \Delta, \text{ contained in some even vertex}\},\\
\Delta_- &:= \{\text{maximal set in } \Delta, \text{ contained in some odd vertex}\},\\
\Upsilon_+ &:= \{\text{maximal set in } \Upsilon, \text{ contained in some even vertex}\},\\
\Upsilon_- &:= \{\text{maximal set in } \Upsilon, \text{ contained in some odd vertex}\}.
\end{aligned}$$

Moreover, let

$$\begin{aligned}
\Delta'_+ &:= \{X \subseteq V \mid X \in \Delta, \text{maximal proper subset of some element in } \Upsilon_+\},\\
\Delta'_- &:= \{X \subseteq V \mid X \in \Delta, \text{maximal proper subset of some element in } \Upsilon_-\},\\
V_+ &:= \{v \in V \mid \{v\} \in \mathrm{even}(H) \text{ or } v \text{ is contained in some even vertex}\},\\
V_- &:= \{v \in V \mid \{v\} \in \mathrm{odd}(H) \text{ or } v \text{ is contained in some odd vertex}\}.
\end{aligned}$$

Then, update (y, z) by

$$y(v) := \begin{cases} y(v) - \epsilon & (v \in V_+),\\ y(v) + \epsilon & (v \in V_-),\\ y(v) & (\text{otherwise}), \end{cases}$$

$$z(\mathcal{L}_U) := \begin{cases} z(\mathcal{L}_U) + 2\epsilon & (U \in \Upsilon_+ \cup (\Delta'_- \setminus \Delta_-)),\\ z(\mathcal{L}_U) - 2\epsilon & (U \in \Upsilon_- \cup (\Delta'_+ \setminus \Delta_+)),\\ z(\mathcal{L}_U) + \epsilon & (U \in (\Delta_+ \setminus \Delta'_+) \cup (\Delta_- \cap \Delta'_-)),\\ z(\mathcal{L}_U) - \epsilon & (U \in (\Delta_+ \cap \Delta'_+) \cup (\Delta_- \setminus \Delta'_-)),\\ z(\mathcal{L}_U) & (\text{otherwise}), \end{cases}$$

where $\epsilon \geq 0$ is the minimum of the following:

$\epsilon_1 = \min\{y(v) \mid v \in V_+\}$; $\epsilon_2 = \min\{z(\mathcal{L}_U)/2 \mid U \in \Upsilon_- \cup (\Delta'_+ \setminus \Delta_+)\}$;

$\epsilon_3 = \min\{z(\mathcal{L}_U) \mid U \in (\Delta_+ \cap \Delta'_+) \cup (\Delta_- \setminus \Delta'_-)\}$;

$\epsilon_4 = \min\{w'(e)/2 \mid e \in \hat{E},\ \partial e \subseteq V_+\}$;

$\epsilon_5 = \min\{w'(e) \mid e \in \hat{E}, \text{ one of } \partial e \text{ belongs to } V_+, \text{ and the other } V \setminus (V_+ \cup V_-)\}$;

$\epsilon_6 = \min\{w'(e) \mid \partial e \subseteq X \text{ for some } X \in \Delta_+ \cup \Delta'_+\}$;

$\epsilon_7 = \min\{w'(a) \mid a \in A,\ \partial^- a \in V_+,\ a \notin \gamma(\mathcal{L}_U) \text{ for any } U \in \Delta_+ \cup \Upsilon_+\}$.

Then, apply one of the following.

Case 1 ($\epsilon = \epsilon_1$): Termination. Deshrink every sets in Ω' and return F.
Case 2 ($\epsilon = \epsilon_2$): Deshrinking. Apply Case 2.1 or 2.2.
Case 2.1 ($\epsilon = z(\mathcal{L}_U)/2$ for $U \in \Upsilon_-$): Deshrinking. Delete U with $\epsilon = z(\mathcal{L}_U)/2$ from Υ, and then go to Step 3.
Case 2.2 ($\epsilon = z(\mathcal{L}_U)/2$ for $U \in \Delta'_+ \setminus \Delta_+$): Deshrinking. Denote the maximal set in Υ containing U by W, and the maximal set in Δ containing U by X. Add $\tilde{U} = W \cup X$ to Υ, define a partition $\mathcal{L}_{\tilde{U}}$ of $\tilde{U}$ by $(\mathcal{L}_U \setminus \{U\}) \cup \{X\}$, and set $z(\mathcal{L}_{\tilde{U}}) = 0$. Then, delete U from Δ and go to Step 3.
Case 3 ($\epsilon = \epsilon_3$). Apply Case 3.1 or 3.2.
Case 3.1 ($\epsilon = z(\mathcal{L}_U)$ for $U \in \Delta_+ \cap \Delta'_+$): Deshrinking and saturation. Denote the set in Υ_+ containing U by W, and the pseudo-vertex containing W by $\hat{W}$. Delete U from Δ, reset $\hat{F} := (\hat{M} \triangle P_{\hat{W}}) \cup \hat{B}$, and label $\hat{W}$ as "saturated." Note that G°_W has a matching forest covering all vertices in G°_W. Delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω', and go to Step 2.
Case 3.2 ($\epsilon = z(\mathcal{L}_U)$ for $U \in \Delta_- \setminus \Delta'_-$). Let $\hat{U}$ be the vertex in $\hat{V}$ containing U, and let $f_1, f_2 \in H$ be the edges incident to $\hat{U}$. If f_1 and f_2 are incident to distinct vertices in G_U, apply Case 3.2.1. Otherwise, apply Case 3.2.2.
Case 3.2.1: Deshrinking and augmentation. Reset $\hat{M} := \hat{M} \triangle P_{\hat{U}}$. Delete U from Δ and each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω', and then go to Step 2.
Case 3.2.2: Deshrinking. Delete U from Δ and then go to Step 3.
Case 4 ($\epsilon = \epsilon_4$). Go to Step 5. (We can execute Step 5.1 or 5.2.)
Case 5 ($\epsilon = \epsilon_5$). Go to Step 5. (We can execute Step 5.3 or 5.4.)
Case 6 ($\epsilon = \epsilon_6$): Saturation. Let $X \subseteq V$ be an element in $\Delta_+ \cup \Delta'_+$ such that contains $e \in E$ with $\epsilon = w'(e)$, and let $\hat{X}$ denote the pseudo-vertex in $\hat{G}$ containing X. Reset $\hat{M} := \hat{M} \triangle P_{\hat{X}}$ and label $\hat{X}$ as "saturated." Note that G°_X has a matching forest covering all vertices in G°_X. Delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω', and then go to Step 2.
Case 7 ($\epsilon = \epsilon_7$). Apply Case 7.1 or 7.2.
Case 7.1 ($\epsilon = w'(a)$ for $a \in \hat{A}$). Go to Step 4.
Case 7.2 ($\epsilon = w'(a)$ for $a \in A_U$ with $U \in \Upsilon_+$): Saturation. Reset $\hat{M} := \hat{M} \triangle P_U$ and label U as "saturated." Note that G°_U has a matching forest covering all vertices in G°_U. Delete each $T \in \Omega'$ with $z(\mathcal{L}_T) = 0$ from Ω', and then then go to Step 2.

We close this section by noting three features of the algorithm. The first one is the time complexity. The bottleneck part is Dual_Update. Procedure Dual_Update is executed $O(n)$ times between consecutive augmentations or saturations. Since augmentations and saturations collectively happen at most n times and Dual_Update takes $O(m)$ time for determining ϵ, the total complexity is $O(n^2 m)$.

Secondly, let us mention the difference from Giles' algorithm [15]. A major difference is in Step 4. In Giles' algorithm, the condition where we go to Step 4.1 is that $\hat{B} \cup \{a\}$ is a branching, which is a sufficient condition of our condition of $R(\hat{B}) \setminus \{v\} \in \hat{\mathcal{F}}^\circ_B$. In other words, there exist cases where our algorithm

executes augmentation but Giles' executes shrinking. In this sense, our algorithm is expected to be practically faster than Giles' algorithm, while they have the same complexity.

The final remark is on a property of the matching forests maintained in the algorithm. Let F be a matching forest which we have at any stage of the algorithm. (Deshrink each element in Ω', if $\Omega \neq \emptyset$.) Then, it is not difficult to see that $w(F) \geq w(F')$ for any matching forest F' with $|R(F')| = |R(F)|$. That is, a matching forest appearing at any stage of the algorithm has the maximum weight among all matching forests with the same root-size.

4 A Faster Algorithm

In this section, we sketch an $O(n^3)$ algorithm for the weighted matching forest problem which incorporates Giles' weighted matching forest algorithm [15] and Gabow's technique for weighted matching [13]. The difference from Algorithm SIMPLE is that we do not maintain the equality subgraph G° explicitly. Instead, we keep the following.

- For each pair Y, Z of disjoint sets in Ω, we keep an edge $e_{YZ} \in E$ connecting Y and Z and minimizing w'. We keep e_{YZ} as lists: for each $Y \in \Omega$, we have a list containing the e_{YZ}. Moreover, for each $Y \in \Omega$, we keep an edge e_Y with $e_Y = e_{YZ}$ for some $Z \in \Omega$ contained in an even (pseudo-)vertex in H and with $w'(e_{YZ})$ minimal. Also, for each pair Y, Z of disjoint sets in Ω, we keep an arc $a_{YZ} \in A$ from Y to Z minimizing w'. We keep a_{YZ} as lists: for each $Z \in \Omega$, we have a list containing the a_{YZ}. Moreover, for each $Z \in \Omega$, we keep an arc a_Z with $a_Z = a_{YZ}$ for some $Y \in \Omega$ and with $w'(a_{YZ})$ minimal.
- For each $X \in \Delta$, we keep $f_X \in E_X$ minimizing w'. We associate a graph G'_X, which is initially the directed cycle shrunk when X is added to Δ.
- For each $U \in \Upsilon$, we keep $b_U \in A_U$ minimizing w'. We associate a graph G'_U, which is initially the odd undirected cycle shrunk when U is added to Υ.

We remark that, in this algorithm, we do not use the branching delta-matroid $\hat{\mathcal{F}}^\circ_B$ (see Step 4 in Algorithm SIMPLE), because determining whether $R(\hat{B}) \setminus \{v\} \in \hat{\mathcal{F}}^\circ_B$ requires $O(m)$ time (decomposition of $(\hat{V}, \hat{A}^\circ)$ into strong components). Instead, we use Giles' condition, whether $\hat{B} \cup \{a\}$ is a branching or not.

The above objects enables us to execute Procedure Dual_Update in $O(n)$ time, by scanning the e_Y, a_Z, f_X b_U, and $z(\mathcal{L}_U)$. Updating the lists after shrinking, deshrinking or forest extension takes $O(n)$ time and updating after augmentation and saturation takes $O(n^2)$ time. Thus, the total time complexity $O(n^3)$.

Acknowledgements

The author is thankful to Satoru Iwata for valuable comments on this research and the simple proof for the matching forest delta-matroid. The author also thanks Yusuke Kobayashi for discussions on simultaneous delta-matroids.

References

1. Bouchet, A.: Greedy Algorithm and Symmetric Matroids. Math. Programming 38, 147–159 (1987)
2. Bouchet, A.: Matchings and Δ-Matroids. Discrete Appl. Math. 24, 55–62 (1989)
3. Chandrasekaran, R., Kabadi, S.N.: Pseudomatroids. Discrete Math. 71, 205–217 (1988)
4. Cunningham, W.H., Marsh III, A.B.: A Primal Algorithm for Optimum Matching. Math. Programming Study 8, 50–72 (1978)
5. Dress, A.W.M., Havel, T.: Some Combinatorial Properties of Discriminants in Metric Vector Spaces. Adv. Math. 62, 285–312 (1986)
6. Dress, A.W.M., Wenzel, W.: A Greedy-Algorithm Characterization of Valuated Δ-matroids. Appl. Math. Lett. 4, 55–58 (1991)
7. Edmonds, J.: Maximum Matching and a Polyhedron with 0,1-Vertices. J. Res. Natl. Bur. Stand. Sect. B 69, 125–130 (1965)
8. Edmonds, J.: Paths, Trees, and Flowers. Canad. J. Math. 17, 449–467 (1965)
9. Edmonds, J.: Optimum Branchings. J. Res. Natl. Bur. Stand. Sect. B 71, 233–240 (1967)
10. Edmonds, J., Giles, R.: A Min-Max Relation for Submodular Functions on Graphs. Ann. Discrete Math. 1, 185–204 (1977)
11. Frank, A.: Covering Branchings. Acta Sci. Math (Szeged) 41, 77–81 (1979)
12. Frank, A., Tardos, É.: Generalized Polymatroids and Submodular Flows. Math. Programming 42, 489–563 (1988)
13. Gabow, H.N.: Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs, Ph.D. thesis, Stanford University (1973)
14. Giles, R.: Optimum Matching Forests I: Special Weights. Math. Programming 22, 1–11 (1982)
15. Giles, R.: Optimum Matching Forests II: General Weights. Math. Programming 22, 12–38 (1982)
16. Giles, R.: Optimum Matching Forests III: Facets of Matching Forest Polyhedra. Math. Programming 22, 39–51 (1982)
17. Keijsper, J.: A Vizing-Type Theorem for Matching Forests. Discrete Math. 260, 211–216 (2003)
18. Lovász, L.: Matroid Matching and Some Applications. J. Combin. Theory Ser. B 28, 208–236 (1980)
19. Murota, K.: Characterizing a Valuated Delta-Matroid as a Family of Delta-Matroids. J. Oper. Res. Soc. Japan 40, 565–578 (1997)
20. Murota, K.: Discrete Convex Analysis. SIAM, Philadelphia (2003)
21. Schrijver, A.: Total Dual Integrality of Matching Forest Constraint. Combinatorica 20, 575–588 (2000)
22. Schrijver, A.: Combinatorial Optimization—Polyhedra and Efficiency. Springer, Heidelberg (2003)
23. Vizing, V.G.: Ob Otsenke Khromaticheskogo Klassa p-grapha (in Russian). Diskretnyĭ 3, 25–30 (1964)
24. Vizing, V.G.: Khromaticheskiĭ Klass Mul'tigrafa (in Russian). Kibernetika 1(3), 29–39 (1965)
25. Wenzel, W.: Δ-Matroids with the Strong Exchange Conditions. Appl. Math. Lett. 6, 67–70 (1993)
26. Wenzel, W.: Pfaffian Forms and Δ-Matroids. Discrete Math. 115, 253–266 (1993)

Fixed-Charge Transportation on a Path: Linear Programming Formulations*

Mathieu Van Vyve

Center for Operations Research and Econometrics and
Louvain School of Management,
Voie du Roman Pays 34, B-1348 Louvain-la-Neuve, Belgium
mathieu.vanvyve@uclouvain.be
http://www.uclouvain.be/core

Abstract. The fixed-charge transportation problem is a fixed-charge network flow problem on a bipartite graph. This problem appears as a subproblem in many hard transportation problems, and has also strong links with the challenging big-bucket multi-item lot-sizing problem. We provide a polyhedral analysis of the polynomially solvable special case in which the associated bipartite graph is a path.

We describe a new class of inequalities that we call "path-modular" inequalities. We give two distinct proofs of their validity. The first one is direct and crucially relies on sub- and super-modularity of an associated set function, thereby providing an interesting link with flow-cover type inequalities. The second proof is by projecting a tight extended formulation, therefore also showing that these inequalities suffice to describe the convex hull of the feasible solutions to this problem. We finally show how to solve the separation problem associated to the path-modular inequalities in $\mathcal{O}(n^3)$ time.

Keywords: mixed-integer programming, lot-sizing, transportation, convex hull, extended formulation.

1 Introduction

In the fixed charge transportation problem (FCT), we are given a set of depots $i \in I$ each with a quantity of available items C_i, and a set of clients $j \in J$ each with a maximum demand D_j. For each depot-client pair (i, j), both the unit profit $q_{i,j}$ of transporting one unit from the depot to the client is known, together with the fixed charge $g_{i,j}$ of transportation along that arc. The goal is

* This paper presents research results of the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian State, Prime Minister's Office, Science Policy Programming. The scientific responsibility is assumed by the author.

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 417–429, 2011.

to find a profit-maximizing transportation program. Problem FCT can therefore be expressed as the following mixed-integer linear program:

$$\max \quad \sum_{i\in I}\sum_{j\in J}(q_{i,j}w_{i,j} - g_{i,j}v_{i,j}), \tag{1}$$

$$\sum_{j\in J} w_{i,j} \le C_i\ , \qquad \forall i \in I\ , \tag{2}$$

$$\sum_{i\in I} w_{i,j} \le D_j\ , \qquad \forall j \in J\ , \tag{3}$$

$$0 \le w_{i,j} \le \min(C_i, D_j)v_{i,j}\ , \qquad \forall i \in I, j \in J\ , \tag{4}$$

$$v \in \{0,1\}^{I\times J}\ , \tag{5}$$

where $w_{i,j}$ is a variable representing the amount transported from depot i to client j and $v_{i,j}$ is the associated binary setup variable.

In this description, the role of clients and depots are interchangeable. Indeed, this problem can be modelled as a bipartite graph in which nodes are either depots or clients and edges between a depot and a client exist if the client can be served from that depot. A standard variant (and indeed a special case) is that in which the demand of each client must be satisfied, in which case the unit profit is usually replaced by a unit cost.

Problem FCT can be considered as a basic problem in supply chain management in its own right, and is also a special fixed-charge network flow problem. However surprisingly few polyhedral results are known for FCT. When there is only one client or one depot, FCT reduces to a single node flow set for which the (lifted) cover and reverse cover inequalities have been described and shown to be effective [11,12,8]. Note that this also implies that FCT is NP-Hard. The flow structure of FCT is similar to that of the Capacitated Facility Location (CFL) problem, but this last problem has fixed cost for opening depots (nodes) as opposed to transportation (edges). Known valid inequalities are essentially flow cover type inequalities [2,5]. FCT also appears to be both a special case and a strong relaxation of the multi-item big-bucket lot-sizing problem [13] for which dual gaps are still typically large in practice [9,14,1]. This actually constitutes the initial motivation of this work.

In this paper, we study the polynomially solvable special case of FCT in which the associated bipartite graph is a path. FCTP is also a relaxation of FCT, and it can therefore be hoped that polyhedral results for FCTP will be helpful in solving the general case.

Let $[k,l]$ denote $\{k, k+1, \ldots, l\}$. For FCTP, we assume that we have $n+1$ depots and clients (or nodes) in total, indexed by the set $[0,n]$. We index depot-client pairs $(i-1,i)$ (or edges) by $i \in [1,n]$. Depots are represented by even

nodes while clients are represented by odd nodes. The problem FCTP can be formulated as the following mixed-integer program:

$$\max \quad \sum_{i=1}^{n} p_j x_j - \sum_{i=1}^{n} f_j y_j \ , \tag{6}$$

$$x_i + x_{i+1} \leq a_i \ , \qquad \forall i \in [1, n-1] \ , \tag{7}$$

$$0 \leq x_i \leq \min(a_{i-1}, a_i) y_i \ , \qquad \forall i \in [1, n] \ , \tag{8}$$

$$y \in \{0, 1\} \ , \qquad \forall i \in [1, n] \ , \tag{9}$$

where x_i for $i \in [1, n]$ is the amount transported between $i-1$ and i, y_i is the setup variable associated with x_i, $a_i > 0$ is the capacity of node i, and p_i and f_i are respectively the unit profit and the fixed cost of transportation between $i-1$ and i. We denote the set of feasible solutions to (7)–(9) by X^{FCTP}.

Program (7)–(9) can easily be recast into the framework introduced by Conforti et al. [4] by operating the change of variables $x'_i = \frac{x_i}{M}$ with M large enough. Continuous variables (nodes) are then linked by a simple path with arcs having alternating directions. This graph trivially only contains a polynomial number of subpaths, so that FCTP admits a compact extended formulation and is therefore polynomially solvable. Parts of Section 2 can be seen as specializing the extended linear-programming formulation of Conforti et al. [6] for FCTP. In addition, in the special case of FCTP we are able to describe a dedicated combinatorial optimization algorithm (see Van Vyve [13]) and to project the extended formulation onto the original variable space (see Sections 2 and 3).

Recently, using the same framework, Di Summa and Wolsey [7] aim at studying the mixed-integer set in which continuous nodes are linked by a bidirected path. This model subsumes FCTP. However they are only able to characterize the convex hull of the set for two special cases that do not subsume FCTP.

The rest of the paper is organized as follows. In Section 2 we give a linear-programming extended formulation for FCTP counting $\mathcal{O}(n^2)$ variables and constraints and a combinatorial characterization of the facet-defining inequalities defining its projection onto the original variable space. We first prove that the proposed formulation is indeed an extended formution of $conv(X^{FCTP})$. This is done by showing that the matrix associated to a subset of the constraints is a network flow matrix, as in [6]. We then prove that the constraint matrix associated with the projection cone of this formulation is a totally unimodular matrix, thereby showing that the coefficients of x_i in facet-defining inequalities of $conv(X^{FCTP})$ have the consecutive ones property. Finally, fixing the coefficients of x_i to one of their $\mathcal{O}(n^2)$ possible values, the separation problem becomes separable, with all subproblems equivalent to matching problems in closely related bipartite graphs. This enables us to give a combinatorial characterization of a family of valid inequalities sufficient to describe $conv(X^{FCTP})$.

In Section 3, we introduce a new class of inequalities for FCTP that we call the "path-modular inequalities". We give a direct proof of their validity relying on sub- and super-modularity properties of an associated set-function. This makes

an intersting link with flow cover inequalities of which the validity for the single-node flow set relies on submodularity of a similar set function. We then show that the path-modular inequalities are in fact identical to the inequalities obtained by projection. We also give an $\mathcal{O}(n^3)$ separation algorithm for the path-modular inequalities.

We conclude by discussing future research on the topic.

Notation. Throughout this paper we use the following notation: $\triangle$ denotes the symetric difference, $[k,l] = \{k, k+1, \ldots, l\}$, $N = [1,n]$, E and O denote the even and odd integers respectively, $\underline{e}_i$ denotes the unit vector with component i equal to 1 and all others components equal to 0, $\underline{1}$ denotes the all ones vector, and $\tilde{y}$ denotes the vector y with odd components being complemented (i.e. replaced by $1 - y_i$).

2 Compact Linear Programming Formulation and Projection

In this section, we give an extended linear programming formulation for FCTP of size $\mathcal{O}(n^2)$ variables and constraints. It can be seen as a specialization for FCTP of the results of Conforti et al. [6]. In addition, we are able to give a combinatorial characterization of the extreme rays of the associated projection cone, and therefore a combinatorial description of inequalities sufficients to describe $conv(X^{FCTP})$.

We start with a definition, a lemma that we state without proof (it can be found in Van Vyve [13]) and a couple of observations.

Definition 1. *For given $j \in [0, n+1]$, let $\bar{\alpha}_{i,j}$ for $i \in [0, n+1]$ be the unique solution of the following system of $n+2$ linear equations in variables x_i for $i \in [0, n+1]$: $x_i + x_{i+1} = a_i$ for $i \in [0,n]$ and $x_j = 0$ (note that x_0 and x_{n+1} are defined here but not in (7)-(9)).*

Lemma 1. *Let x be an extreme point of X^{FCTP}. Then x_i takes its value in the set $\{\bar{\alpha}_{i,j}\}_{j=0}^{n+1}$.*

Observation 1. *$\bar{\alpha}_{i,j}$ satisfy the two following properties:*

$$\bar{\alpha}_{i,j} + \bar{\alpha}_{i+1,j} = a_i \ , \qquad \forall i \in [0,n], j \in [0, n+1] \ , \tag{10}$$

$$\bar{\alpha}_{i,j} = \begin{cases} \bar{\alpha}_{i,0} + \bar{\alpha}_{0,j} \ i \in E \ , \\ \bar{\alpha}_{i,0} - \bar{\alpha}_{0,j} \ i \in O \ . \end{cases} \tag{11}$$

Let $(j_0, j_1, \ldots, j_{\bar{m}})$ be a permutation of a subset of $[0, n+1]$ that removes duplicate entries and sorts $\{\bar{\alpha}_{1,j}\}_{j=0}^{n+1}$ in increasing order: $\bar{\alpha}_{1,j_0} < \bar{\alpha}_{1,j_1} < \ldots < \bar{\alpha}_{1,j_{\bar{m}}}$. For fixed $i \in [1,n]$, it follows from (11) that the same permutation removes duplicate entries and sorts $\{\bar{\alpha}_{i,j}\}_{j=0}^{n+1}$ in increasing (resp. decreasing) order for i odd (resp. even). Defining

$$\bar{\beta}_{i,k} = \begin{cases} \bar{\alpha}_{i,j_k} & \text{if } i \in [1,n] \cap O, \ k \in [0, \bar{m}] \ , \\ \bar{\alpha}_{i,j_{k-1}} & \text{if } i \in [1,n] \cap E, \ k \in [1, \bar{m}+1] \ . \end{cases}$$

and $\gamma_k = \bar{\beta}_{1,k} - \bar{\beta}_{1,k-1} > 0$ for $k \in [1, \bar{m}]$, (11) implies that

$$\bar{\beta}_{i,k} - \bar{\beta}_{i,k-1} = \gamma_k, \text{for } i \in [1,n] \cap O,\ k \in [1, \bar{m}] ,$$
$$\bar{\beta}_{i,k} - \bar{\beta}_{i,k+1} = \gamma_k, \text{for } i \in [1,n] \cap E,\ k \in [1, \bar{m}] .$$

Consider now the following formulation in which the intended meaning is that $z_{i,k} = 1$ if x_i takes a value at least $\bar{\beta}_{i,k}$ and 0 otherwise.

$$x_i = \bar{\beta}_{i,0} + \sum_{k=1}^{\bar{m}} \gamma_k z_{i,k} , \qquad \forall i \in O , \tag{12}$$

$$x_i = \bar{\beta}_{i,\bar{m}+1} + \sum_{k=1}^{\bar{m}} \gamma_k z_{i,k} , \qquad \forall i \in E , \tag{13}$$

$$y_i \geq z_{i,k} , \qquad \forall i \in N, k \in [1, \bar{m}] : \bar{\beta}_{i,k} > 0 , \tag{14}$$

$$z_{i,k} + z_{i+1,k} \leq 1 , \qquad \forall i \in N, k \in [1, \bar{m}] , \tag{15}$$

$$z_{i,k} = 0 , \qquad \forall i \in N, k \in [1, \bar{m}] : \bar{\beta}_{i,k} > \min(a_{i-1}, a_i) , \tag{16}$$

$$z_{i,k} = 1 , \qquad \forall i \in N, k \in [1, \bar{m}] : \bar{\beta}_{i,k} \leq 0 , \tag{17}$$

$$z_{i,k} \geq 0 , \qquad \forall i \in N, k \in [1, \bar{m}] . \tag{18}$$

The next proposition shows that this is a correct formulation of FCTP.

Proposition 1. *For any $(x, y) \in \mathbb{R}^n \times \mathbb{Z}^n$, $(x, y) \in X^{FCTP}$ if and only if there exists z such that (x, y, z) is feasible in (12)-(18).*

Proof. We first show that for any $(x, y) \in X^{FCTP}$ there exists z such that (x, y, z) is feasible in (12)-(18). Assuming wlog that (x, y) is an extreme point of X^{FCTP}, let $z_{i,k} = 1$ if x_i takes a value at least $\bar{\beta}_{i,k}$ and 0 otherwise. That this choice of z satisfies constraints (12)-(18) is clear except for (15). Suppose $i \in O$ (the case $i \in E$ is similar). Because $\bar{\beta}_{i,k} + \bar{\beta}_{i+1,k} = \bar{\alpha}_{i,j_k} + \bar{\alpha}_{i+1,j_{k-1}} = \bar{\alpha}_{i,j_k} + \bar{\alpha}_{i+1,j_k} + \gamma_k = a_i + \gamma_k > a_i$, the relation $z_{i,k} + z_{i+1,k} > 1$ would imply $x_i + x_{i+1} > a_i$, a contradiction.

We now show the converse. First let us show that constraints (7) are implied by (12),(13),(15). Suppose i is odd (the other case is similar). By (10) and definition of γ, we know that $\beta_{i,0} + \beta_{i+1,\bar{m}+1} = (\beta_{i+1,\bar{m}} + \beta_{i,\bar{m}}) + (\beta_{i,0} - \beta_{i,\bar{m}}) = a_i - \sum_{k=1}^{\bar{m}} \gamma_k$. Therefore we can write

$$\begin{aligned} x_i + x_{i+1} &= \bar{\beta}_{i,0} + \sum_{k=1}^{\bar{m}} \gamma_k z_{i,k} + \bar{\beta}_{i+1,\bar{m}} + \sum_{k=1}^{\bar{m}} \gamma_k z_{i+1,k} \\ &= a_i - \sum_{k=1}^{\bar{m}} \gamma_k + \sum_{k=1}^{\bar{m}} \gamma_k (z_{i,k} + z_{i+1,k}) \\ &\leq a_i - \sum_{k=1}^{\bar{m}} \gamma_k + \sum_{k=1}^{\bar{m}} \gamma_k = a_i . \end{aligned}$$

We finally show that (x, y, z) feasible in (12)-(18) and $x_i > 0$ implies $y_i > 0$. Observe that because of (17) and the fact that $\bar{\beta}_{i,k} = 0$ for some k, (12) can

be rewritten as $x_i = \sum_{k=1:\bar{\beta}_{i,k}>0}^{\bar{m}} \gamma_k z_{i,k}$. Because of (18) and $\gamma_k > 0$ for all k, $x_i > 0$ if and only if $z_{i,k} > 0$ for some k. This implies $y_i > 0$ using (14). Hence (12)-(18) is a correct formulation of FCTP. ■

Proposition 2. *Formulation* (12)-(18) *is a linear programming extended formulation of* $conv(X^{FCTP})$.

We show that extreme points of (12)-(18) are integral. By the change of variables $z'_{i,j} = -z_{i,j}$ and $y'_i = -y_i$ for i even and $z'_{i,j} = z_{i,j}$ and $y'_i = y_i$ for i odd, each constraint (14) and (15) becomes a bound on a difference of two variables. The matrix associated to this modified constraint system is therefore a network flow matrix. The result follows. ■

A linear programming extended formulation automatically leads to a characterization of the convex hull of the solutions in the original space of variables through projection. Indeed, testing if a point (x^*, y^*) satisfying $y^* \leq \underline{1}$ belongs to $conv(X^{FCTP})$ is equivalent to testing whether the following LP in variables z is feasible:

$$
\begin{aligned}
\max \quad & 0 \,, \\
& \sum_{k \in K_i} \gamma_k z_{i,k} = x_i^* \,, \qquad \forall i \in N \,, && (\Delta_i) \\
& z_{i,k} \leq y_i^* \,, \qquad \forall i \in N, k \in K_i \,, && (\delta_{i,k}) \,, \\
& z_{i,k} + z_{i+1,k} \leq 1 \,, \qquad \forall i \in [1, n-1], k \in K_i \cap K_{i+1} \,, && (\rho_{i,k}) \\
& z_{i,k} \geq 0 \,, \qquad \forall i \in N, k \in K_i \,,
\end{aligned}
$$

where $K_i = \{k \in [1, \bar{m}] : 0 < \bar{\beta}_{i,k} \leq \min(a_{i-1}, a_i)\}$. Through LP duality, this is equivalent to testing whether the following LP is bounded:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} x_i^* \Delta_i + \sum_{i=1}^{n} \sum_{k \in K_i} y_i^* \delta_{i,k} + \sum_{i=1}^{n-1} \sum_{k \in K_i \cap K_{i+1}} \rho_{i,k} \,, \\
& \gamma_k \Delta_i + \delta_{i,k} + \rho_{i-1,k} + \rho_{i,k} \geq 0 \,, \qquad \forall i \in N, k \in K_i \,, \\
& \rho_{i,k} = 0 \,, \qquad \forall i \notin [1, n-1] \text{ or } k \notin K_i \cap K_{i+1} \,, \\
& \delta, \rho \geq 0 \,.
\end{aligned}
$$

Dividing the constraint by γ_k and rescaling $\delta_{i,k}$ and $\rho_{i,k}$ by γ_k, one obtains the equivalent LP

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} x_i^* \Delta_i + \sum_{i=1}^{n} \sum_{k \in K_i} \gamma_k y_i^* \delta_{i,k} + \sum_{i=1}^{n-1} \sum_{k \in K_i \cap K_{i+1}} \gamma_k \rho_{i,k} \,, && (19) \\
& \Delta_i + \delta_{i,k} + \rho_{i-1,k} + \rho_{i,k} \geq 0 \,, \qquad \forall i \in N, k \in K_i \,, && (20) \\
& \rho_{i,k} = 0 \,, \qquad \forall i \notin [1, n-1] \text{ or } k \notin K_i \cap K_{i+1} \,, && (21) \\
& \delta, \rho \geq 0 \,. && (22)
\end{aligned}
$$

This is true if (x^*, y^*) satisfies

$$\sum_{i=1}^{n} x_i^* \Delta_i + \sum_{i=1}^{n} \sum_{k \in K_i} \gamma_k y_i^* \delta_{i,k} + \sum_{i=1}^{n-1} \sum_{k \in K_i \cap K_{i+1}} \gamma_k \rho_{i,k} \geq 0$$

for all extreme rays of the cone associated to the last constraint system. We will characterize sufficient inequalities to describe the polyhedron $conv(X^{FCTP})$ by characterizing these extreme rays. Because $x^*, y^*, \gamma, \delta, \rho \geq 0$, extreme rays with negative cost satisfy $\Delta \leq 0$. Hence we can normalize rays by assuming $\Delta_i \geq -1$ for all $i \in N$.

A first observation is that $\Delta_i < 0$ for consecutive i's (otherwise the ray is the sum of two other rays). The following result is less trivial.

Proposition 3. *The matrix associated to the constraint system* (20) *is totally unimodular.*

Proof. Variable $\delta_{i,k}$ appears only in one constraint and can therefore be neglected. We prove the result by proving that for any subset J of the columns of the matrix B under consideration, there exists a partition (J^-, J^+) of J such that $\sum_{j \in J^+} b_{i,j} - \sum_{j \in J^-} b_{i,j} \in \{-1, 0, 1\}$ for all rows i.

If none of the variables Δ_i belong to J then the remaining matrix satisfies the consecutive ones property and is TU. So we can assume the contrary and consider columns associated to Δ_i belonging to J in increasing order of i: $i_1 < i_2 < \ldots$. We assign Δ_{i_1} to J^+. Then we assign Δ_{i_j} to the same partition as $\Delta_{i_{j-1}}$ if the parity of i_j and i_{j-1} are different and to the other partition if the parity is the same (in particular, consecutive columns are assigned to the same partition).

Note that having partitioned columns associated to Δ, the problem becomes separable in k. Consider $\rho_{i',k}$ belonging to J and let $j^<$ be the highest index such that $i_{j^<} \leq i'$ and let $j^>$ be the lowest index such that $i_{j^>} > i'$. In other words, $i_{j^<}$ and $i_{j^>}$ are the two closest columns Δ_i before and after i' belonging to J. At least one of them exists. We assign $\rho_{i,k}$ to

- the same partition as $\Delta_{i_{j^<}}$ if $i_{j^<}$ and i' have a different parity,
- the opposite partition to that of $\Delta_{i_{j^<}}$ if $i_{j^<}$ and i' have the same parity,
- the same partition as $\Delta_{i_{j^>}}$ if i' and $i_{j^>}$ have the same parity,
- the opposite partition to that of $\Delta_{i_{j^>}}$ if i' and $i_{j^>}$ have a different parity.

Observe that these rules cannot be contradictory in case both $j^<$ and $j^>$ exist because of the choosen partitioning of Δ_i.

We claim that this partitioning satisfies the desired property for row (20) for any given i, k. If Δ_i does not belong to J, then $\rho_{i-1,k}$ and $\rho_{i,k}$ are assigned to opposite partitions and the property holds. If Δ_i belongs to J, then $\rho_{i-1,k}$ and $\rho_{i,k}$ are both assigned to the opposite partition to that of Δ_i and again the property holds. ■

Corollary 1. *Normalized extreme rays of negative cost of* (19)-(22) *satisfy* $\Delta_i = -1$ *if* $i \in [l, l']$ *and* 0 *otherwise for some* $l, l' \in N$.

Therefore we can assume without loss of generality Δ fixed accordingly, and analyze optimal solutions of the LP (19)-(22) under this assumption. The following characterization will be sufficient for our purposes.

Proposition 4. *For given (x^*, y^*) and fixed Δ according to Corollary 1, the set of optimal solutions to (19)-(22) is the same for all vectors y^* that admit the same ordering when sorted in decreasing order of $\tilde{y}^*$. When y^* is such that this ordering is unique, the optimal solution is unique as well.*

Proof. Observe that for fixed Δ, the LP (19)-(22) is separable in k. Furthermore, for given each k, the problem is of the form:

$$\min \quad \sum_{i=1}^{n} y_i^* \delta_i + \sum_{i=1}^{n-1} \rho_i \ , \tag{23}$$

$$\delta_1 + \rho_1 \geq 1 \ , \tag{24}$$

$$\delta_i + \rho_{i-1} + \rho_i \geq 1 \ , \qquad \forall i \in [2, n-1] \ , \tag{25}$$

$$\delta_n + \rho_{n-1} \geq 1 \ , \tag{26}$$

$$\rho_i = 0 \ , \qquad \forall i \in Q \ , \tag{27}$$

$$\delta, \rho \geq 0 \ , \tag{28}$$

where Q can be choosen to represent constraints (21). From Proposition 3, we know that optimal solutions can be assumed to be integral. From a graphical perspective, we have an undirected path of which each node i must be covered either by itself at cost y_i^* or by one of its incident edges at cost 1. Note that as $0 \leq y_i^* \leq 1$, we can assume that in optimal solutions all inequalities (24)-(26) will be tight. Indeed, if inequality i is not tight and $\Delta_i > 0$, we can decrease it by 1 without increasing the objective. If inequality i is not tight and $\Delta_i = 0$, then ρ_{i-1} or ρ_i is positive. Suppose ρ_i. Then we can decrease ρ_i by 1 and increase y_{i+1}^* by 1 without increasing the objective.

Therefore the same problem can be modelled as the more classical perfect matching problem in the following bipartite graph. The node set is $V = I \cup I'$ where $I = I' = [1, \bar{n}]$ and we index nodes in I (resp. I') by i (resp. i'). The edge set E includes edges (i, i') if $i = i'$ with cost y_i^* and edges $(i, i+1)$ and $(i', i'+1)$ for all $i, i' \in [1, \bar{n}-1] \setminus Q$ with cost $\frac{1}{2}$. Note that this graph is bipartite but not under the usual partition $E \subseteq I \times I'$. The two problems are equivalent because if edge $(i, i+1)$ is in the matching, then the edge $(i', i'+1)$ for $i' = i$ must also be in the matching and together they cost 1.

Elementary cycles in this bipartite graph are of the form $i, i+1, \ldots, j, j', j'-1, \ldots, i'$ and can therefore be unambigously denoted by $C_{i,j}$ for $i < j$. Consider a given perfect matching M of the graph (V, E) just defined. $C_{i,j}$ is an alternating cycle with respect to M if and only if the four following conditions hold:

(i) M does not contain an edge (l, l') with $i < l < j$,
(ii) if either (i, i') or (j, j') but not both belongs to M, then i and j are of the same parity,

(iii) if either both (i, i') and (j, j') or none of them belong to M, then i and j are of different parity.
(iv) there is no $l \in Q$ with $i \leq l < j$.

Let $C_{i,j}$ be such an alternating cycle and consider the perfect matching M' obtained by taking the symetric difference $M' = M \triangle C_{i,j}$. Denoting the cost of matching M by $c(M)$, the structure of the graph implies that:

$$c(M') = C(M) + \begin{cases} 1 - y_i^* - y_j^* & \text{if both } (i,i') \text{ and } (j,j') \text{ belong to } M, \\ y_i^* + y_j^* - 1 & \text{if neither } (i,i') \text{ nor } (j,j') \text{ belong to } M, \\ y_i^* - y_j^* & \text{if } (j,j') \text{ belongs to } M \text{ but not } (i,i'), \\ y_j^* - y_i^* & \text{if } (i,i') \text{ belongs to } M \text{ but not } (j,j'). \end{cases}$$

Combining this with the characterization of an alternating cycle above, we obtain that the set of optimal solutions (matchings) to (23)-(28) will be the same for all vectors y^* that admit the same orderings when sorted in decreasing order of $\tilde{y}^*$. When this ordering is unique the optimal matching is unique as well as taking the symetric difference with any elementary alternating cycle will strictly increase its cost.

For fixed Δ, subproblems k of (19)-(22) will only differ by Q in constraint (27). Hence the same is true for optimal solutions of (19)-(22). ■

Corollary 2. *Together with bounds $x_i \geq 0$ and $y_i \leq 1$ for $i \in N$, the following family of valid inequalities is sufficient to describe the convex hull of X^{FCTP}:*

$$\sum_{i=l}^{l'} x_i \leq \tau(\mathcal{L}) + \sum_{i=l}^{l'} \sigma(i, \mathcal{L}) y_i \ , \tag{29}$$

where $l, l' \in N$, $l \leq l'$, $\mathcal{L}$ is a permutation of $[l, l']$ and $\tau(\mathcal{L}) = \sum_{i=1}^{n-1} \sum_{k \in K_i \cap K_{i+1}} \gamma_k \rho_{i,k}$ and $\sigma(i, \mathcal{L}) = \sum_{k \in K_i} \gamma_k \delta_{i,k}$ for the unique optimal solution (δ, ρ) of (19)-(22) obtained when $\mathcal{L}$ is the unique permutation that sorts $\tilde{y}^*$ in decreasing order and Δ is fixed at $\Delta_i = -1$ for $i \in [l, l']$ and 0 otherwise.

3 Path-Modular Inequalities

In this section we derive a new family of inequalities that we call path-modular inequalities. Before showing that they are equivalent to inequalities (29), we give a more direct and insightful proof of their validity for FCTP. This proof relies on submodularity and supermodularity properties of the following set function. For a given set $S \subseteq [1, n]$ and vector $a \in \mathbb{R}_+^{n+1}$, let $\phi(S, a)$ be defined as

$$\begin{aligned} \phi(S, a) = \max \quad & \sum_{i=1}^{n} x_j \ , \\ & (7) \ , \\ & x_i \geq 0 \ , \qquad \forall i \in S \ , \\ & x_i = 0 \ , \qquad \forall i \in N \setminus S \ . \end{aligned}$$

For notational convenience, we will sometimes omit the argument a in $\phi(S,a)$ when we unambiguously refer to the original input data of the problem. Let $\rho_i(S) = \phi(S+i) - \phi(S)$ be the increment function of i at S. Clearly $\rho_i(S)$ is always nonnegative, but it is neither globally submodular ($\rho_i(S) \geq \rho_i(T)$ for all $S \subset T, i \notin T$) nor supermodular ($\rho_i(S) \leq \rho_i(T)$ for all $S \subset T, i \notin T$).

We now define the path-modular inequalities. Let L be a subinterval $[l,l']$ of N, let $L = O_L \cup E_L$ be the partition of L into odd and even numbers, and let $\mathcal{L} = (j_1, j_2, \ldots, j_{|L|})$ be a permutation of L. Let $O_L^{j_k} = \{j_1, \ldots, j_k\} \cap O_L$ and let $E_L^{j_k} = \{j_1, \ldots, j_{k-1}\} \cap E_L$. We call the the following inequality the $(l,l',\mathcal{L})$-path-modular inequality:

$$\sum_{i \in L} x_i \leq \phi(O_L) + \sum_{i \in E_L} \rho_i(O_L \cup E_L^i \setminus O_L^i) y_i - \sum_{i \in O_L} \rho_i(O_L \cup E_L^i \setminus O_L^i)(1-y_i) \ . \quad (30)$$

The following proposition essentially characterizes certain vectors y at which a given path-modular inequality is tight.

Proposition 5. *Let an interval $L=[l,l']$ and a permutation $\mathcal{L}=(j_1, j_2, \ldots, j_{|L|})$ of L be given. For each $k \in [0,|L|]$, there exists a point $(x^k, y^k) \in X^{FCTP}$ which is tight for the corresponding $(l,l',\mathcal{L})$-path-modular inequality, satisfying*

$$y^k = \sum_{i \in O_L} \underline{e}_i \triangle \sum_{1}^{k} \underline{e}_{j_k}$$

and at which the inequality reduces to $\sum_{i \in Y^k} x_i \leq \phi(Y^k)$ where $Y^k = \{i \in N : y_i^k = 1\}$.

Proof. The proof is by induction on k. For $k=0$, $Y^0 = O_L$ and the path modular inequality reduces to $\sum_{i \in O_L} x_i \leq \phi(O_L)$. By definition of ϕ, there exists x such that this inequality is satisfied at equality.

So let us assume that the proposition is true for $k-1$ and $k>0$. If j_k is even, then the inequality reduces to $\sum_{i \in Y^k} x \leq \phi(Y^{k-1}) + \rho_{j_k}(Y^{k-1}) = \phi(Y^{k-1} + j_k) = \phi(Y^k)$. If j_k is odd, then the inequality reduces to $\sum_{i \in Y^k} x \leq \phi(Y^{k-1}) - \rho_{j_k}(Y^{k-1} - j_k) = \phi(Y^k - j_k) = \phi(Y^k)$. By definition of ϕ, there exists x such that this inequality is satisfied at equality. ∎

In particular the previous proposition shows that all path-modular inequalities are tight at points with exactly either all odd or all even edges open.

The following proposition essentially tells us that ϕ exhibits supermodularity when we keep on opening edges of the same parity as the edge i under consideration. Conversely, it tells us also that ϕ exhibits submodularity when we keep on opening edges of the opposite parity compared to the edge i under consideration. Note that, broadly speaking, this is also the case for flow cover inequalities: in that case each edge is at an odd distance from any other one, so that the associated max-flow function is completely submodular. The technical and lengthy proof of this proposition can be found in [13].

Proposition 6. *Let $S \subset T \subseteq N$ and $i \in N \setminus T$ be given.*

(i) If $(T \setminus S) \subseteq E$ and $i \in E$, then $\rho_i(T) \geq \rho_i(S)$.
(ii) If $(T \setminus S) \subseteq O$ and $i \in O$, then $\rho_i(T) \geq \rho_i(S)$.
(iii) If $(T \setminus S) \subseteq E$ and $i \in O$, then $\rho_i(T) \leq \rho_i(S)$.
(iv) If $(T \setminus S) \subseteq O$ and $i \in E$, then $\rho_i(T) \leq \rho_i(S)$.

We are now ready to prove the validity of the path-modular inequalities.

Proposition 7. *The path-modular inequalities (30) are valid for X^{FCTP}.*

Proof. Let a feasible point $(x, y) \in X^{FCTP}$ be given with $Y = \{i \in N : y_i = 1\}$. Let $L \subseteq N$ be a set of consecutive integers and let a permutation $\mathcal{L} = (j_1, j_2, \ldots, j_{|L|})$ of L be given. We show that the point (x, y) satisfies the corresponding path-modular inequality.

Let $\bar{E}^i_L = E^i_L \cap Y$ and $\bar{O}^i_L = O^i_L \cap Y$. The following relations hold

$$\begin{aligned}
\sum_{i \in L} x_i &\leq \phi(Y) \\
&= \phi(O_L) - \sum_{i \in O_L \setminus Y} \rho_i(O_L \cup \bar{E}^i_L \setminus \bar{O}^i_L) + \sum_{i \in E_L \cap Y} \rho_i(O_L \cup \bar{E}^i_L \setminus \bar{O}^i_L) \\
&\leq \phi(O_L) - \sum_{i \in O_L \setminus Y} \rho_i(O_L \cup \bar{E}^i_L \setminus O^i_L) + \sum_{i \in E_L \cap Y} \rho_i(O_L \cup \bar{E}^i_L \setminus O^i_L) \\
&\leq \phi(O_L) - \sum_{i \in O_L \setminus Y} \rho_i(O_L \cup E^i_L \setminus O^i_L) + \sum_{i \in E_L \cap Y} \rho_i(O_L \cup E^i_L \setminus O^i_L) \\
&= \phi(O_L) - \sum_{i \in O_L} \rho_i(O_L \cup E^i_L \setminus O^i_L)(1 - y_i) + \sum_{i \in E_L} \rho_i(O_L \cup E^i_L \setminus O^i_L) y_i \,,
\end{aligned}$$

where the first inequality is by definition of ϕ, the first equality is by definition of the increment function ρ and the fact that the set argument of each term (ordered according to the order $(j_1, j_2, \ldots, j_{|L|})$) differs from the previous term by exactly the element being incremented or decremented, the second inequality is an application of Corollary 6 (ii) for the first term and (iv) for the second term, the third inequality is an application of Corollary 6 (iii) for the first term and (i) for the second term, and finally the last equality holds because the added terms in the sum are null. ∎

We now prove that the path-modular inequalities are sufficient to describe $conv(X^{FCTP})$ by showing that they are the same as the inequalities (29) obtained by projection.

Proposition 8. *Together with bounds $x_i \geq 0$ and $y_i \leq 1$ for $i \in N$, the path-modular inequalities are sufficient to describe the convex hull of X^{FCTP}.*

Proof. Let $l, l' \in N$, $l \leq l'$ and a permutation $\mathcal{L}$ of $[l, l']$ be given. Consider any of the $|l' - l + 2|$ points (x^k, y^k) of Proposition 5 for the ordering $(j_1, j_2, \ldots, j_n) = \mathcal{L}$. Such a point lies on the boundary of $conv(X^{FCTP})$, and therefore a separation

algorithm that maximizes the violation will output a valid inequality that is tight at this point. Consider now the LP (19)-(22) with Δ fixed at $\Delta_i = -1$ for $l \leq i \leq l'$ and 0 otherwise. This LP actually determines a valid inequality of the form $\sum_{i=l}^{l'} x_i \leq \pi_0 + \sum_{i=l}^{l'} \pi_i y_i$ maximizing the violation. Now observe that the permutation $\mathcal{L}$ sorts $\tilde{y}^k$ in decreasing order for any k. Therefore the corresponding inequality (29) is tight at (x^k, y^k) for any k. By Proposition 5, this is also the case for the path-modular inequality associated to $(l, l', \mathcal{L})$. As these $|l' - l + 2|$ points are affinely independent, these two inequalities are identical. ■

We now turn to the question of separating a point (x^*, y^*) with $x_i^* \geq 0$ and $y_i^* \leq 1$ for all $i \in N$ from the polyhedron defined by the path-modular inequalities. It follows directly from Proposition 4 that the ordering maximizing the violation sorts $\tilde{y}^*$ in decreasing order. Moreover this ordering is independent of L.

It is explained in [13] how preprocessing that can be carried out $\mathcal{O}(n^2)$ operations makes it possible to compute each coefficient of a given path-modular inequality in constant time. The next result follows.

Proposition 9. *The separation problem associated with the path-modular inequalities can be solved in $\mathcal{O}(n^3)$ time. The separation problem associated with the path-modular inequalities with $|L| \leq k$ can be solved in $\mathcal{O}(n^2 + k^2 n)$ time.*

4 Conclusion

This paper is a polyhedral analysis of the Fixed Charge Transportation problem on Paths (FCTP). We describe a new family of valid inequalities for FCTP that we call path-modular inequalities. We give a direct proof of their validity relying on sub- and super-modularity properties of an associated set function. We show that they are sufficient to describe the convex hull of solutions to FCTP by projecting a linear-programming extended formulation. We also show how to separate path-modular inequalities in $\mathcal{O}(n^3)$ time.

In an extended version of this paper [13] we also characterize extreme points of FCTP, we give a combinatorial optimization algorithm, we present an alternative extended formulation, we report on computational experiment in solving FCTP using the different formulations and cutting planes presented, and show that a substantial number of facets of FCT are in fact path-modular inequalities of path-relaxations of FCT.

In our view, this work can be pursued in three main directions. The first one is trying to generalize path-modular inequalities to other graph structures. In particular, it would be nice to be able to describe a family of inequalities that subsumes path-modular (paths) and simple flow-cover inequalities (stars). The second one is to study how this work can help in going forward in the study of doubly-linked mixing sets [7]. The third one is to use the present work to better solve general fixed charge transportation problems.

Acknowledgements

The author is grateful to L.A. Wolsey for commenting on an earlier version of this text.

References

1. Akartunal, K., Miller, A.J.: A Computational Analysis of Lower Bounds for Big Bucket Production Planning Problems (2007), Optimization Online, http://www.optimization-online.org/DB_FILE/2007/05/1668.pdf
2. Aardal, K.: Capacitated Facility Location: Separation Algorithms and Computational Experience. Mathematical Programming A 81, 149–175 (1998)
3. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice Hall, Inc., Englewood Cliffs (1993)
4. Conforti, M., Di Summa, M., Eisenbrand, F., Wolsey, L.A.: Network formulations of mixed-integer programs. Mathematics of Operations Research 34, 194–209 (2009)
5. Carr, R., Fleischer, L., Leung, V., Phillips, C.: Strengthening integrality gaps for capacitated network design and covering problems. In: Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 106–115 (2000)
6. Conforti, M., Wolsey, L.A., Zambelli, G.: Projecting an extended formulation for mixed-integer covers on bipartite graphs. Mathematics of Operations Research 35, 603–623 (2010)
7. Di Summa, M., Wolsey, L.A.: Mixing sets linked by bidirected paths. CORE Discussion Paper 2010/61, Louvain-la-Neuve (2010)
8. Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P.: Lifted flow cover inequalities for mixed 0-1 integer programs. Mathematical Programming A 85, 439–467 (1999)
9. Jans, R., Degraeve, Z.: Improved lower bounds for the capacitated lot sizing problem with setup times. Operations Research Letters 32, 185–195 (2004)
10. Krarup, J., Bilde, O.: Plant location, set covering and economic lotsizes: an $O(mn)$ algorithm for structured problems. In: Collatz, L. (ed.) Optimierung bei Graphentheoretischen und Ganzzahligen Probleme, pp. 155–180. Birkhauser Verlag, Basel (1977)
11. Padberg, M.W., van Roy, T.J., Wolsey, L.A.: Valid linear inequalities for fixed charge problems. Operations Research 33, 842–861 (1985)
12. van Roy, T.J., Wolsey, L.A.: Solving mixed integer programming problems using automatic reformulation. Operations Research 33, 45–57 (1987)
13. Van Vyve, M.: Fixed-charge transportation on a path: optimization, LP formulations and separation. CORE Discussion Paper 2010/68, Louvain-la-Neuve (2010)
14. Van Vyve, M., Wolsey, L.A.: Approximate extended formulations. Mathematical Programming B 105, 501–522 (2006)

Author Index